

工业自动化技术与应用丛书

# LabVIEW 快速入门与 工程实例

曹丽英 主 编

秦 波 杨 斌 副主编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

## 内 容 简 介

LabVIEW 是目前广泛使用的虚拟仪器程序设计软件。本书介绍 LabVIEW 的基础知识、设计方法和工程应用,注重理论联系实践。主要内容包括 LabVIEW 的基本概念、软件安装、用户界面、编程环境、数据操作、程序结构、数据类型、图形化显示、文件操作、数字计算和信号分析。此外,结合作者在教学和研究工作中对该软件的使用经验,精选了 LabVIEW 工程应用实例。

本书提供了大量的实例和图示说明,内容丰富、论述简洁,突出内容的系统性和实用性,力求使读者能够快速入门并达到基本应用水平。

本书主要面向 LabVIEW 的初中级用户,可作为高等院校工科类本科生的教材,也可作为相关工程技术人员和科研工作者的参考书。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。  
版权所有,侵权必究。

## 图书在版编目(CIP)数据

LabVIEW 快速入门与工程实例 / 曹丽英主编.——北京:电子工业出版社,2017.4

(工业自动化技术与应用丛书)

ISBN 978-7-121-31124-6

I. ①L… II. ①曹… III. ①软件工具—程序设计 IV. ①TP311.56

中国版本图书馆 CIP 数据核字(2017)第 057521 号

策划编辑:陈韦凯

责任编辑:万子芬 特约编辑:徐 宏

印 刷:

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1 092 1/16 印张:19.25 字数:493 千字

版 次:2017 年 4 月第 1 版

印 次:2017 年 4 月第 1 次印刷

印 数:3 500 册 定价:55.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888, 88258888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式:bjewk@163.com。

# 前 言

虚拟仪器实际上是一个按照仪器需求组织起来的数据采集系统。虚拟仪器研究涉及的基础理论主要是计算机数据采集和数字信号处理，目前，在这一领域内使用较为广泛的计算机软件是美国 NI 公司的 LabVIEW。

虚拟仪器的起源可以追溯到 20 世纪 70 年代，当时计算机测控系统在国防、航天等领域已经有了相当好的发展。PC 的出现使仪器的计算机化成为可能，甚至在 Microsoft 公司的 Windows 诞生之前，NI 公司已经在 Macintosh 计算机上推出了 LabVIEW 2.0 以前的早期版本。

对虚拟仪器和 LabVIEW 长期、系统、有效的研究开发使得 NI 公司成为业界公认的权威。LabVIEW 是图形化开发环境语言，又称 G 语言，结合了图形化编程方式的高性能与灵活性，以及专为测试、测量与自动化控制应用设计的高性能模块及其配置功能，能为数据采集、仪器控制、测量分析与数据显示等各种应用提供必要的开发工具。

本书在编写过程中详细介绍了学习 LabVIEW 所要注意的问题，使读者更加深刻地理解各种函数与 VI。本书通过“知识点—实例—知识点—实例”的形式介绍全书内容，以理论构建主干，以实例填补枝蔓，内容丰富全面，充满实战性，有利于读者全面掌握本书所介绍的内容，锻炼实际操作能力。

本书主要面向 LabVIEW 的初中级用户，可作为高等院校相关专业的教材和参考书，也可供有关工程技术人员和软件工程师参考。

本书由内蒙古科技大学机械工程学院曹丽英担任主编，秦波、杨斌担任副主编，内蒙古科技大学范丽荣、刘文婧、孟智慧、朱文艳、许少峰，以及内蒙古建筑职业技术学院王栓巧参编。

各章节具体分工如下：第 1、4 章由刘文婧编写，第 2 章、第 3 章（3.2~3.4 节）、第 10 章由范丽荣编写，第 5 章、第 8 章和第 12 章（12.1~12.3 节）由孟智慧编写，第 6 章、第 12 章（12.4 节）由曹丽英编写，第 7 章和附录 C 由朱文艳编写，第 9 章和第 12 章（12.5 节）由许少峰编写，第 11 章由王栓巧编写，第 3 章（3.1 节）、第 13 章（13.1 节、13.3 节）和附录 A、B 由秦波编写，第 13 章（13.2、13.4、13.5 节）由杨斌编写。全书由曹丽英编写大纲，并负责统稿和定稿。

本书由内蒙古科技大学汪建新教授主审。研究生史兴华、杨左文和张弘玉在资料收集、整理方面做了大量工作，特表示感谢！

本书得到了 2015 年度内蒙古科技大学教材建设项目的支持，特表示感谢！

由于编者水平有限，书中难免存在疏漏和不足，欢迎广大读者批评指正，编者将不胜感激。

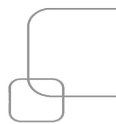
主 编  
2017 年 2 月



# 目 录

第 1 章 LabVIEW 入门知识	1
1.1 LabVIEW 2013 概述	1
1.1.1 LabVIEW 的基本概念	1
1.1.2 LabVIEW 的功能特点	1
1.1.3 LabVIEW 的发展历程	2
1.1.4 LabVIEW 的工程应用	3
1.2 LabVIEW 的安装	4
1.2.1 安装环境	4
1.2.2 安装过程	4
1.3 LabVIEW 的使用	5
第 2 章 LabVIEW 的用户界面	6
2.1 前面板	6
2.1.1 数值、布尔、字符串与路径	7
2.1.2 数组	10
2.1.3 簇	11
2.1.4 图形	12
2.1.5 三维图形	16
2.1.6 极坐标图	24
2.2 程序框图	25
2.3 连线板	27
2.4 对象的选择与删除	28
2.5 设置对象的位置关系	28
2.5.1 对齐关系	28
2.5.2 分布对象	29
2.5.3 改变对象在窗口中的前后次序	29
2.5.4 组合与锁定对象	30
2.5.5 网格布局	30
第 3 章 LabVIEW 的编程环境	31
3.1 启动界面和文件管理	31

3.1.1	新建 VI	31
3.1.2	编辑 VI 图标	32
3.1.3	保存 VI	33
3.1.4	新建文件	33
3.1.5	创建项目	35
3.2	控件选板	36
3.2.1	新式控件	36
3.2.2	经典控件	43
3.2.3	银色控件	47
3.2.4	系统控件	51
3.2.5	Express 控件	53
3.3	函数选板、工具选板及选板可见性设置	56
3.4	项目浏览器	59
第 4 章	程序框图设计基础	62
4.1	程序框图结构	62
4.2	数学函数与 VI	64
4.2.1	数值函数	64
4.2.2	初等与特殊函数	68
4.2.3	函数快捷命令	69
4.3	VI 的设计	70
4.3.1	创建 VI 前面板	70
4.3.2	创建程序框图	73
4.3.3	对象连接	74
4.3.4	运行 VI	76
4.3.5	设置图标	77
4.4	调试 VI	79
4.5	子 VI	82
4.6	性能和内存信息	87
4.7	搜索控件、VI 和函数	88
第 5 章	数据操作	90
5.1	概述	90
5.2	数据类型	90
5.2.1	数字型	90
5.2.2	布尔型	93
5.2.3	枚举类型	94
5.2.4	时间类型 (Time Stamp)	97
5.2.5	变体数据类型 (Variant)	98
5.2.6	局部变量和全局变量	99
5.3	数据运算	101



5.3.1 算术运算 .....	101
5.3.2 关系运算 .....	102
5.3.3 逻辑运算 .....	104
5.3.4 表达式节点 .....	104
<b>第 6 章 程序结构 .....</b>	<b>108</b>
6.1 顺序结构 .....	108
6.2 循环结构 .....	111
6.2.1 For 循环 .....	111
6.2.2 While 循环 .....	118
6.3 层次结构 .....	121
6.3.1 条件结构 .....	121
6.3.2 事件结构 .....	125
6.3.3 使能结构 .....	129
6.4 公式节点 .....	130
6.4.1 复杂公式的实现 .....	130
6.4.2 文本编辑语言的实现 .....	132
6.5 结构函数传递函数 .....	134
6.5.1 反馈节点 .....	134
6.5.2 移位寄存器 .....	136
6.6 定时循环 .....	139
6.6.1 定时循环和定时顺序结构 .....	139
6.6.2 配置定时循环和定时顺序结构 .....	142
6.7 变量 .....	147
6.7.1 局部变量 .....	148
6.7.2 全局变量 .....	149
<b>第 7 章 数组/字符串/簇和矩阵 .....</b>	<b>151</b>
7.1 数组 .....	151
7.1.1 数组的组成与创建 .....	151
7.1.2 使用循环创建数组 .....	152
7.1.3 数组函数 .....	152
7.2 簇 .....	155
7.2.1 簇的组成与创建 .....	155
7.2.2 簇数据的使用 .....	156
7.3 字符串 .....	161
7.3.1 字符串控件 .....	163
7.3.2 表格和树形控件 .....	163
7.3.3 字符串函数 .....	164
7.4 矩阵 .....	169
<b>第 8 章 图形化显示 .....</b>	<b>170</b>



8.1 波形图 .....	170
8.2 波形图表 .....	174
8.3 XY 图 .....	176
<b>第 9 章 文件操作 .....</b>	<b>181</b>
9.1 文件操作的 VI 和函数 .....	181
9.2 文件 I/O 实例 .....	190
<b>第 10 章 属性节点和方法节点 .....</b>	<b>193</b>
10.1 属性节点 .....	193
10.2 调用节点 .....	198
10.3 实例 .....	198
<b>第 11 章 数学计算 .....</b>	<b>200</b>
11.1 基本数学函数 .....	202
11.2 线性代数 .....	203
11.3 曲线拟合 .....	204
11.4 插值 .....	206
11.5 数值积分与数值微分 .....	208
11.6 概率与统计 .....	210
11.7 最优化 .....	211
11.8 常微分方程 .....	213
11.9 几何 .....	216
11.10 公式解析 .....	217
<b>第 12 章 波形分析 .....</b>	<b>219</b>
12.1 波形生成 .....	220
12.2 波形调理 .....	221
12.3 波形测量 .....	222
12.4 信号分析 .....	224
12.4.1 信号的时域分析 .....	224
12.4.2 信号的频域分析 .....	233
12.5 信号变换 .....	240
12.5.1 信号变换相关的 VI .....	241
12.5.2 信号变换举例 .....	242
<b>第 13 章 LabVIEW 的应用实例 .....</b>	<b>248</b>
13.1 轧机齿轮箱双通道振动数据采集系统设计 .....	248
13.1.1 设计目的 .....	248
13.1.2 设计要求 .....	248
13.1.3 设计过程 .....	248
13.2 风机齿轮箱监测与分析系统设计 .....	256



13.2.1 设计目的 .....	256
13.2.2 系统的硬件构成 .....	256
13.2.3 系统软件设计 .....	257
13.3 矿井通风机轴承振动和温度监测系统 .....	262
13.3.1 系统的总体设计方案 .....	262
13.3.2 前面板设计 .....	263
13.3.3 程序框图设计 .....	265
13.4 温度检测系统设计 .....	273
13.4.1 设计目的 .....	273
13.4.2 系统的硬件构成 .....	273
13.5 压力测试系统 .....	280
13.5.1 设计目的 .....	280
13.5.2 系统前面板设计 .....	280
13.5.3 系统程序框图设计 .....	282
附录 A LabVIEW 13.0 快捷键汇总 .....	288
附录 B 公式节点和表达式节点中的内建函数 .....	294
附录 C 公式节点和表达式节点中的数学运算符 .....	296
参考文献 .....	297



# 第 1 章 LabVIEW 入门知识

## 1.1 LabVIEW 2013 概述

### 1.1.1 LabVIEW 的基本概念

LabVIEW 是实验室虚拟仪器集成环境（Laboratory Virtual Instrument Engineering Workbench）的简称，是美国国家仪器公司（National Instruments, NI）的创新软件产品，也是目前应用最广、发展最快、功能最强的图形化软件开发集成环境，又称 G 语言。与 Visual C++、Delphi、Perl 等基于文本型程序代码的编程语言不同，LabVIEW 采用图形模式的结构框图构建程序代码，因而，在使用这种语言编程时，基本上不写程序代码，取而代之的是用图标、连线构成的流程图。它尽可能地利用了开发人员、科学家、工程师所熟悉的术语、图标和概念。因此，LabVIEW 是一个面向最终用户的工具，它可以增强用户构建科学和工程系统的能力，提供了实现仪器编程和数据采集系统的便捷途径。

作为虚拟仪器的领头羊，美国国家仪器公司有涵盖几乎各个行业测试测量应用的产品线，其中的软件代表产品就是 LabVIEW 和 LabWindows/CVI，而 LabVIEW 的最新版本已经升级到 LabVIEW 2014，开发系统适用于 Windows、Mac OS 和 Linux 操作系统，NI 公司虚拟仪器软件开发产品包括四大类：基本版、完整版、专业版和 NI 开发者套件。

- 基本版：适于高效的数据采集和仪器控制的图形化开发。
- 完整版：为测试、信号处理和自动化增加了内容丰富的分析功能。
- 专业版：适于高级开发者或开发团队的解决方案。
- NI 开发者套件：包括 LabVIEW 专业版，附加工具包和季度更新。

在全球工业领域，例如汽车、通信、航空、半导体、电子设计生产、过程控制和生物学等领域，LabVIEW 都能为工程解决实际问题，提高应用程序的开发效率，从目前应用情况来看，LabVIEW 已经涵盖了从研发、测试、生产到服务的产品开发的各个阶段。

### 1.1.2 LabVIEW 的功能特点

LabVIEW 的主要特点可概括如下。

- 图形化的仪器编程环境，它使用“所见即所得”的可视化技术建立人机界面，使用大多数工程师所熟悉的数据流程图式的语言编写程序，被誉为“工程师和科学家的语言”。
- 内置的程序编译器，使运行速度加快。
- 灵活的程序调试手段，可以在源代码中设置断点，单步运行，在数据流上设置探

针，加亮执行。其中最具特色的是“加亮执行”和“设置探针”，前者用于跟踪程序运行过程中的数据流，后者用于在程序运行过程中在线显示数据。

- LabVIEW 提供了从底层 VXI、GPIB、串口及数据采集板的控制子程序到大量的仪器驱动程序，从基本的功能函数到高级分析库，几乎涵盖了仪器设计所需要的所有函数。
- 支持多种系统平台，平台之间的程序可以直接进行移植。
- 提供 CLF (Call Library Function) 功能和 CIN (Code Interface Node) 功能，可以直接调用其他软件平台编译的模块。
- 支持 TCP/IP、DDE 等功能。

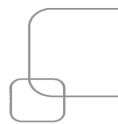
此外，LabVIEW 带有附加的软件包，如磁盘管理、自动测试、控制与仿真、信号处理、图形获得与处理、数值分析工具等。

### 1.1.3 LabVIEW 的发展历程

LabVIEW 从 1986 年发明至今，已推出了数个不同版本，可以支持多个目前流行的操作系统，LabVIEW 的主要发展历程如下所述。

- 1983 年 4 月，LabVIEW 开发系统在美国德克萨斯州奥斯汀研制成功，主要是为仪器系统的开发者提供一套快捷建立、检测和修改仪器系统的图形软件系统。
- 1986 年 5 月，NI 公司推出了 LabVIEW Beta 测试版。
- 1986 年 10 月，NI 公司正式发布了 LabVIEW 1.0 for Macintosh 版本，该版本是解释型和单色的，一问世便引起了仪器工业的变革。
- 1990 年 1 月，LabVIEW 2.0 版本问世，LabVIEW 2.0 是编译型的版本，增加了彩色的性能，它提供了图形编译功能，使得 LabVIEW 中的 VI (虚拟仪器) 运行速度可以与编译 C 语言的运行速度相媲美。
- 1992 年 8 月，支持 Sun Solaris 工作站和 PC 的 LabVIEW 版本面世。
- 1993 年 1 月，LabVIEW 3.0 版本开发完成，同时给用户提供了一个应用系统生成器，使得 LabVIEW 的 VI 变成一个可以独立运行的程序。
- 1998 年 2 月，LabVIEW 5.0 版本面世，该版本是 LabVIEW 历史上的又一个里程碑，为多核设备预先设置了多线程功能，还进行了包括可编程设计的控制面板、用户定义控制、应用程序发行等重大改进。
- 2003 年，LabVIEW 7 Express 和 LabVIEW 7 系列开始推向市场，在 LabVIEW 7 系列中，引入了新的数据类型——动态数据类型 (Dynamic Data Type)，并增加了 LabVIEW PDA 和 LabVIEW FPGA 等各种不同的功能模块。
- 2005 年，LabVIEW 8 版本面世，该版本具有分布式、智能化的优异特性。
- 2006 年，20 周年纪念版 LabVIEW 8.20 面世。LabVIEW 8.2.1 是中文版，也是 LabVIEW 诸多版本中唯一的中文版。
- 2007 年 8 月，LabVIEW 8.5 版本面世。
- 2008 年 8 月，LabVIEW 8.6 版本发布，它提供了支持多核处理器、现场可编辑门阵列 (FPGAs) 编程等最新技术。





- 2010 年 8 月, LabVIEW 2010 版本发布。
- 2011 年 8 月, LabVIEW 2011 版本发布, 这版包含了新的多核 NI CompactRIO 控制器及性能强大的射频向量信号分析器之一的 NI PXIe-5665。LabVIEW 2011 还支持内置在最新的 Microsoft.NET 框架的组件, 并且基于用户的反馈新增了多项新特性。
- 2012 年 8 月, LabVIEW 2012 版本发布。
- 2013 年 8 月, LabVIEW 2013 版本发布。
- 2014 年 8 月, LabVIEW 2014 版本发布。
- 2015 年 8 月, LabVIEW 2015 版本发布。
- 2016 年 8 月, LabVIEW 2016 版本发布。

从 NI 的 LabVIEW 版本号可以看出:

- (1) 系列号 5、6、7、8 表示新的系列, 软件结构或功能可能有重大改进 (付费升级)。
- (2) 版本号 5.x、6.x、7.x、8.x 表示软件有新的内容或比较大的改进 (付费升级)。
- (3) 版本号 5.x.x、6.x.x、7.x.x、8.x.x 表示软件较上个版本进行了修补 (免费升级)。

### 1.1.4 LabVIEW 的工程应用

由于 LabVIEW 可以用来创建通用的应用程序, 因此被称为一种通用的编辑语言。但是它在测试、测量和自动化等领域具有更大的优势, 因为 LabVIEW 提供了大量的工具与函数用于数据采集、分析、显示和存储, 同时还提供了大量常用于自动化测试测量领域的图形控件, 这使得用户可以在数分钟内完成一套完整的从仪器连接、数据采集到分析、显示和存储的自动化测试测量系统。因此它被广泛地应用于汽车、通信、航空、半导体、电子设计生产、过程控制和生物医学等各个领域, 涵盖了从研发、测试、生产到服务的产品开发所有阶段。NI 网站有上千个应用案例供读者参考: <http://www.ni.com/solutions/>。今天欧美的许多高校非计算机专业的学生选修 G 语言并用它开发应用软件的人数已经超过 C 语言等文本语言。近年来我国高校 G 语言教学实践正在迅速展开。

LabVIEW 不仅可以用来快速搭建小型自动化测试测量系统, 还可以用来开发大型的分布式数据采集与控制系统。

在美国 Lawrence Livermore 国家实验室, 一个花费 2000 万美金的极为复杂的飞秒激光切割系统就是基于 LabVIEW 开发的。该系统中, 4 台 Windows NT 工作站用网络连接起来, LabVIEW 用来给激光提供测量、控制和自动定序, 同时作为半熟练操作人员的高层用户界面。几乎安装了所有类型的 I/O 硬件: DAQ、GPIB、串行、远程控制 SCXI、VME/VXI 以及 IMAQ 成像。由于这个项目的极端重要性, 项目采取了正式的软件质量保证过程。软件开发总共用了 4 年, 创建了约 600 个 VI。

在北京正负电子对撞机二期工程北京谱仪慢控制系统中, 大约有 30 种物理量共 7000 多点的现场数据点需要实时采集控制和分析记录, 该系统由 8 台计算机与 2 台服务器组成, 8 台计算机不间断地采集来自于十几种硬件设备的数据, 并将其分析、汇总和本地显示。2 台服务器实现数据的存储和网络发布, 供科学家们随时随地获得或控制探测器的状态。该大型分布式监控系统的上层软件完全基于 LabVIEW 及其 DSC 模块实现, 共创建了约 300 个 VI。

基于 LabVIEW 实现的最大的系统是 Honeywell-Measurex 公司由 Dirk Demol 领导的小组开发的 MxProline。它是一流的分布式过程控制系统, 95% 的代码都是用 LabVIEW 编写的。该系统使用了 5000 个以上的 VI, 可以处理超过 10 万个变量 (包括物理 I/O 和

计算值)。

## 1.2 LabVIEW 的安装

### 1.2.1 安装环境

LabVIEW 可以安装在 Mac Os、Linux、Windows 2000/XP/Vista 等不同的操作系统或同一操作系统的不同版本上。针对不同的操作系统，在安装 LabVIEW 2013 时对系统的配置要求也不同，用户在安装 LabVIEW 2013 前需对计算机系统的软硬件环境配置有一定的了解。

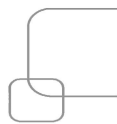
### 1.2.2 安装过程

LabVIEW 的安装十分简单，只需运行安装光盘中的 setup 程序，按照屏幕提示，一步步地选择必要的安装选项即可完成。整个系统安装时间取决于硬件平台和选择的安装选项。LabVIEW 所有文件约占 900MB 的硬盘空间。LabVIEW 13.0 的安装程序界面如图 1.1 所示。



图 1.1 LabVIEW 13.0 安装程序界面

为了控制 VXI、GPIB 和 DAQ 设备，在 LabVIEW 2013 系统安装完成后，如果需要，还



必须安装专门的仪器驱动和 VISA 库函数，它们一般不随 LabVIEW 2013 系统光盘，例如 DAQ 卡的驱动程序，对应于 LabVIEW 2013 的 DAQmx 为 NIDAQ921-1，需从 NI 网站免费下载或从购买 DAQ 采集卡时 NI 公司提供的数张驱动程序光盘获取，需单独安装。

## 1.3 LabVIEW 的使用

LabVIEW 作为目前国际上优秀的编译型图形化编程语言，把复杂、烦琐和费时的语言编程简化成用菜单或图标提示的方法选择功能（图形），使用通过线条把各种功能连接起来的简单图形编程方式。LabVIEW 中编写的框图程序很接近程序流程图，因此，只要把程序流程图画好，程序也就基本编好了。

LabVIEW 中的程序查错不需要先编译，若存在语法错误，LabVIEW 会马上告诉用户。只要用鼠标单击两三下，用户就可以快速查到错误的类型、原因以及错误的准确位置，这个特性在程序较大的情况下使用特别方便。

LabVIEW 中的程序调试方法同样令人称道，程序测试的数据探针工具最具典型性。用户可以在程序调试运行的时候，在程序的任意位置插入任意多的数据探针，检查任意一个中间结果。增加或取消一个数据探针，只需要单击两下鼠标就行了。

同传统的编程语言相比，采用 LabVIEW 图形编程方式可以节省大约 60% 的程序开发时间，并且其运行速度几乎不受影响。

除了具备其他语言所提供的常规函数功能外，LabVIEW 中还集成了大量的生成图形界面的模板、丰富实用的数值分析、数字信号处理功能以及多种硬件设备驱动功能（包括 RS232、GPIB、VXI、数据采集板卡和网络等）。另外，免费提供的几十家仪器厂商的数百种源码仪器级驱动程序，可为用户开发仪器控制系统节省大量的编程时间。

## 第2章 LabVIEW 的用户界面

### 2.1 前 面 板

在 LabVIEW 中开发的程序都被称为 VI（虚拟仪器），其扩展名为 .vi。所有的 VI 都包括前面板（Front Panel）、程序框图（Block Diagram）如图 2.1 所示以及图标（Icon）3 部分。

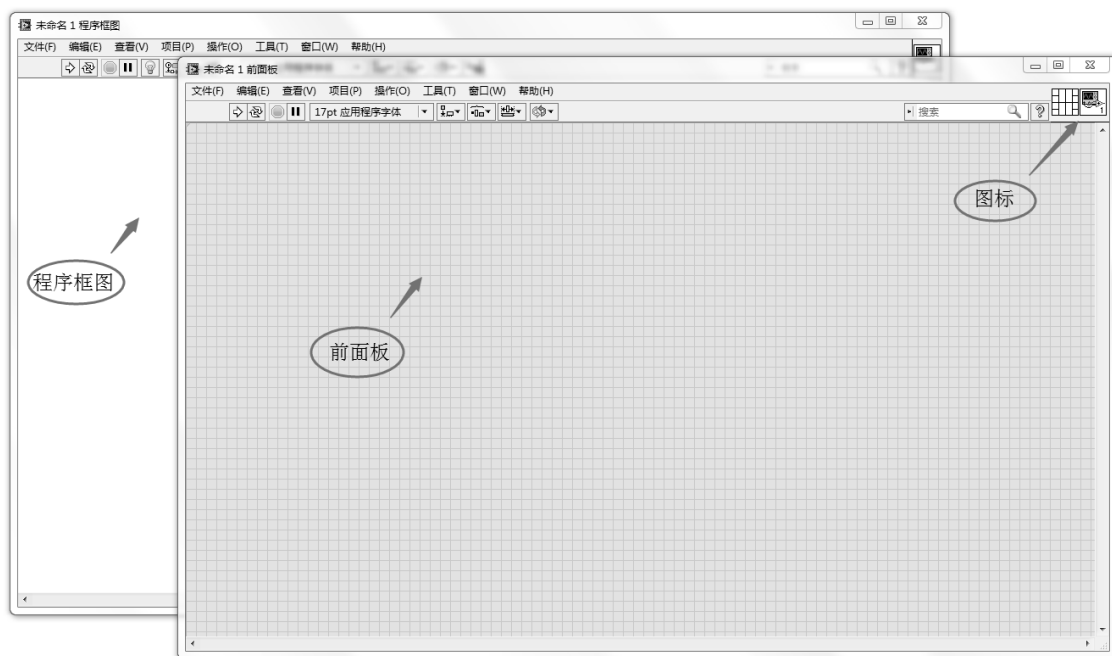


图 2.1 LabVIEW 前面板和程序框图

前面板是 VI 的交互式用户界面，外观和功能都类似于传统仪器面板，用户的输入数据通过前面板传给框图，计算和分析结果也在前面板上以数字、图形、表格等各种不同方式显示出来。

前面板上的各种控件根据输入、输出功能可分为输入控件（Control）和显示控件（Indicator）两类，前者一般用于接受用户输入，包括编辑框、按钮、开关、旋钮等，后者则一般用于显示输出结果，包括图形、指示灯、指示框等，如图 2.2 所示。

前面板的设计不单单是将选中的控件放到仪器中就结束的，前面板的设计不仅是将选中的控件放到仪器中。在内，需要设置控件属性；在外，需要将控件排列美观。

前面板中的控件本身也不是杂乱无章的，是有一定规律的，控件根据功能需要按照类型进行选择。

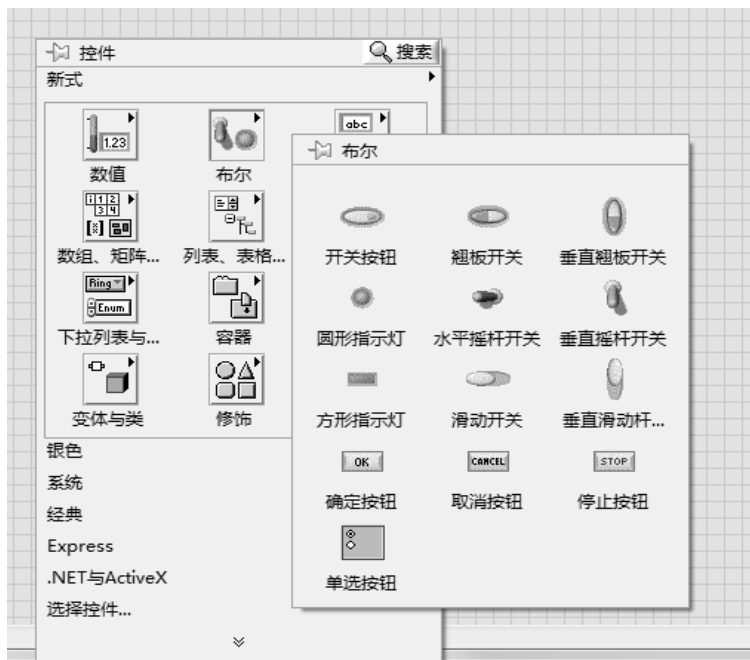


图 2.2 控件

### 2.1.1 数值、布尔、字符串与路径

数值、布尔、字符串与路径均包括输入、输出控件，如图 2.3 所示，在控件选板中选择的控件显示如图 2.4 所示。

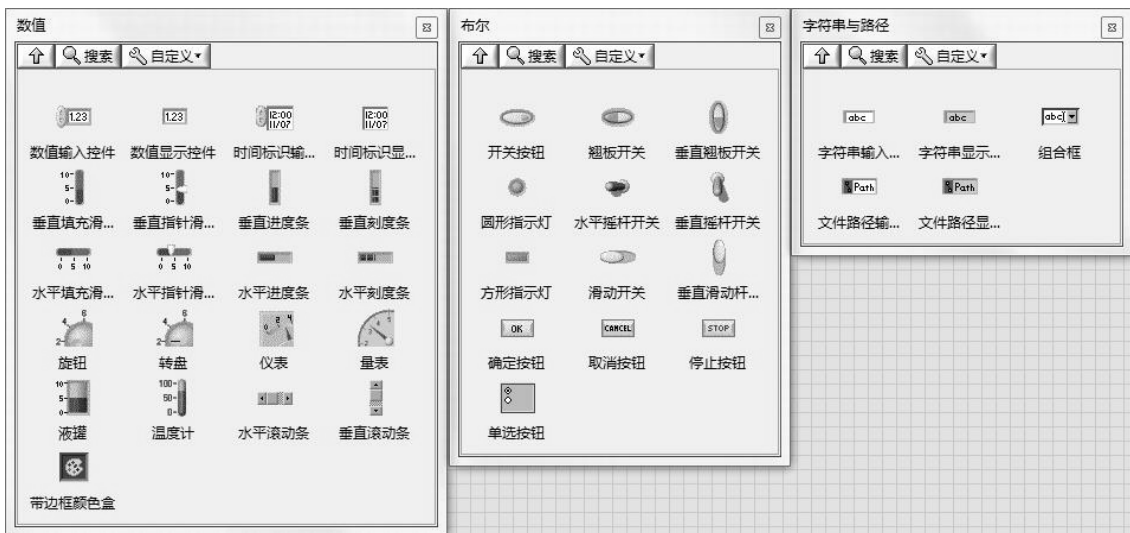


图 2.3 数值、布尔、字符串与路径控件



图 2.4 控件显示

## 1. 数值型控件

数值型控件的数值对象用于输入和显示数值。

### 1) 数值控件

数值控件是输入和显示数值的最简单方式。这些前面板对象可以在水平方向上调整大小，以显示更多位数。使用下列方法可以改变数值控件的值。

- 用操作工具或标签工具单击数字显示框，然后通过键盘输入数字。
- 用操作工具单击数值控件的递增或递减箭头。
- 使用操作工具或标签工具将光标放置于需要改变的数字右边，然后在键盘上按向上或向下箭头键。
- 在默认状态下，LabVIEW 的数字显示和存储与计算器类似。数值控件一般最多显示 6 位数字，超过 6 位数字则自动转换为科学计数法表示。右键单击数值对象并从快捷菜单中选择格式与精度，打开数值属性对话框的格式与精度选项卡，从中配置 LabVIEW 在切换到科学计数法之前所显示的数字位数。

### 2) 滑动杆控件

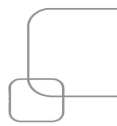
滑动杆控件是带有可读的数值对象。滑动杆控件包括垂直和水平滑动杆、液罐和温度计，可使用下列方法改变滑动杆控件的值。

- 使用操作工具单击或拖曳滑块至新的位置。
- 与数值控件中的操作类似，在数字显示框中输入新数据。

滑动杆控件可以显示多个值。右键单击该对象，在快捷菜单中选择添加滑块，可添加更多滑块。带有多个滑块控件的数据类型为包含各个数值的簇。

### 3) 滚动条控件

与滑动控件相似，滚动条控件用于滚动数据的数值对象。滚动条控件有水平和垂直两种滚动条。使用操作工具单击或拖曳滑块至一个新的位置，单击递增和递减箭头，或单击滑块和箭头之间的空间都可以改变滚动条的值。



#### 4) 旋转型控件

旋转型控件包括旋钮、转盘、量表和仪表。旋转型对象的操作与滑动杆控件相似，都是带有刻度的数值对象。可使用下列方法改变旋转型控件的值。

- 用操作工具单击或拖曳指针至一个新的位置。
- 与数值控件中的操作类似，在数字显示框中输入新数据。

旋转型控件可显示多个值。右键单击该对象，选择添加指针，可添加新指针。带有多个指针的控件的数据类型为包含各个数值的簇。

#### 5) 时间标识控件

时间标识控件用于向程序框图发送或从程序框图获取时间和日期值，如图 2.5 所示。



图 2.5 时间标识控件

可用下列方法改变时间标识控件的值。

- 单击“时间/日期浏览”按钮，显示“设置时间和日期”对话框，如图 2.6 所示。



图 2.6 “设置时间和日期”对话框

- 右键单击该控件并从快捷菜单中选择“数据操作—设置时间和日期”，显示“设置时间和日期”对话框。
- 右键单击该控件，从快捷菜单中选择“数据操作—设置为当前时间”。

## 2. 布尔型控件

布尔型控件可用于创建按钮、开关和指示灯。

布尔输入控件有 6 种机械动作。自定义布尔对象，可创建运行方式与现实仪器类似的前

面板，快捷菜单可用于自定义布尔对象的外观，以及模拟单击这些对象时它们的运行方式。

单选按钮控件向用户提供一个列表，每次只能从中选择一项。如允许不选任何项，右键单击该控件，然后在快捷菜单中选择允许不选，该菜单项旁边将出现一个勾选标志。单选按钮控件为枚举类型，所以可用单选按钮控件选择条件结构中的条件分支。

### 3. 字符串与路径控件

#### 1) 字符串控件

操作工具或标签工具可用于输入或编辑前面板上字符串控件中的文本。在默认状态下，新文本或经改动的文本在编辑操作结束之前不会被传至程序框图。运行时，单击面板的其他位置，切换到另一窗口，单击工具栏上的“确定”按钮，或按<Enter>键，都可以结束编辑状态。在主键区按<Enter>键将输入回车符。右键单击字符串控件为其文本选择显示类型，如以密码形式显示或十六进制数显示。

#### 2) 组合框控件

组合框控件可用于创建一个字符串列表，在前面板上可循环浏览该列表。组合框控件类似于文本型或菜单型下拉列表控件，但是，组合框控件是字符串型数据，而下拉列表控件是数值型数据。

#### 3) 路径控件

路径控件用于输入或返回文件或目录的地址（Windows 和 Mac OS 系统）。如允许运行时拖放，则可以从 Windows 浏览器中拖曳一个路径、文件夹或文件放置在路径控件中。

路径控件与字符串控件的工作原理类似，但 LabVIEW 会根据用户使用操作平台的标准句法将路径按一定格式处理。

## 2.1.2 数组

在程序设计语言中，数组是一种常用的数据结构，是相同数据类型数据的集合，是一种存储和组织相同类型数据的方式。与其他程序设计语言一样，LabVIEW 中的数组是数值型、布尔型、字符串型等多种数据类型中的同类数据的集合，在前面板的数组对象往往由一个盛放数据的容器和数据本身构成。

数组是由同一类型数据元素组成的大小可变的集合。当有一串数据需要处理时，它们可能是一个数组，当需要频繁地对一批数据进行绘图时，使用数组将会事半功倍，数组作为组织绘图数据的一种机制是十分有用的，如解答线性方程。在 VI 中使用数组能够压缩框图代码，并且由于具有大量的内部数组函数和 VI，使得代码开发更加容易。

可以通过以下两步来实现数组输入控件或数组显示控件的创建。

(1) 从控件选板中选取数组、矩阵控件，将其中的数组拖入前面板中，如图 2.7 所示。



(2) 将需要有效数据对象拖入数组框，切记此要点，如果不分配数据类型，该数据将显示为带空括号的黑框。

如图 2.8 所示，数组 1 为未分配数据类型的数组，数组 2 为分配了数据类型的数组，所以此时边框显示为橙黄色。

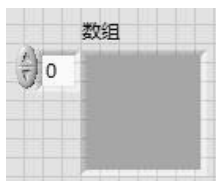


图 2.7 创建数组的第一步



图 2.8 创建数组的第二步

在数组框图的左端或左上角为数组的索引值，显示在数组左边方框中的索引值对应数组中第一个可显示的元素，通过索引值的组合可以访问数组中的每一个元素。LabVIEW 中的数组与其他编程语言相比更灵活，任何一种数据类型的数据（数组本身除外）都可以组成数组。其他的编程语言如 C 语言，在使用一个数组时，必须首先定义数组的长度，但 LabVIEW 却不必如此，它会自动确定数组的长度。在内存允许的情况下，数组中每一维的元素最多可以达到  $2^{31}-1$  个。数组中元素的数据类型必须完全相同，如都是无符号 16 位整数，或全为布尔型等。当数组中有  $n$  个元素时，元素的索引号从 0 开始，到  $n-1$  结束。

### 2.1.3 簇

簇是 LabVIEW 中的一个比较特别的数据类型，它可以将几种不同的数据类型集中到一个单元中形成一个整体，类似于 C 语言中的结构。

簇通常用于将出现在框图上的有关数据元素分组管理。因为簇在框图中仅用唯一的连线表示，所以可以减少连线混乱和子 VI 需要的连接器端子个数。使用簇有积极的效果，可以将簇看作一捆连线，其中每条连线表示簇不同的元素。在框图上，只有当簇具有相同元素类型、相同元素数量和相同元素顺序时，才可以将簇的端子连接。

簇和数组的异同：簇可以包含不同类型的数据，而数组仅可以包含相同的数据类型，簇和数组中的元素都是有序排列的，但访问簇中元素最好是通过释放方法同时访问其中的部分或全部元素，而不是通过索引一次访问一个元素，簇和数组的另一差别是簇具有固定的大小。簇和数组的相似之处是二者都是由输入控件或输出控件组成的，不能同时包含输入控件和输出控件。

簇的创建类似于数组的创建。首先在控制选板的“数组、矩阵与簇”子选板中创建簇的框架，如图 2.9 所示。

然后向簇框架中添加所需要的元素，并且可以根据需要更改簇和簇中各元素的名称，如图 2.10 所示。

一个簇变为输入控件簇或显示控件簇取决于放进簇的第一个元素，若放进簇框架中的第一个元素是布尔控件，那么后来给簇添加的任何元素都将变成输入对象，簇变成了输入控件簇，并且当从任何簇元素的快捷菜单中选择转换为输入控件或显示控件时，簇中的所有元素

都将发生变化。



图 2.9 创建簇的第一步



图 2.10 创建簇的第二步

在簇框架上单击右键弹出快捷菜单，在菜单的“自动调整大小”中的 3 个选项可以用来调整框架的大小以及簇元素的布局，选择匹配大小选项调整簇框架的大小，以适合所包含的所有元素；水平排列选项水平压缩排列所有元素；垂直排列选项垂直压缩排列所有元素。

簇的元素有一定的排列顺序，簇元素按照其放入簇中的先后排序，而不是按照簇框架内的物理顺序排序，簇框架中的第一个对象标记为 0，第 2 个标记为 1，依次排列。在簇中删除元素时，剩余元素的顺序将自动调整，在簇的解除捆绑和捆绑函数中，簇顺序决定了元素的显示顺序。如果要访问簇中的单个元素，必须记住簇的顺序，因为簇中的单个元素都是按顺序访问的。

在前面板上，从簇边框上右键单击，在弹出的快捷菜单中选择“重新排序簇中控件”选项，可以检查和改变簇内元素的顺序，此时图中的工具编程了一组新按钮，簇的背景也有变化，连光标也改变为簇排序光标，选择“重新排序簇中控件”后，簇中每一个元素右下角都出现了并排的框、白框和黑框。白框指出该元素在簇顺序中的当前位置，黑框指出在用户改变顺序的新位置，在此顺序改变前，白框和黑框中的数字是一样的，用簇排序光标单击某个元素，该元素在簇顺序中的位置就会变成顶部工具条显示的数字，单击“×”按钮后可恢复到以前的排列顺序。

应该注意簇顺序的重要性，使用簇时应当遵循的原则是：在一个高度交互的面板中，不要把一个簇既作为输入又作为输出。

## 2.1.4 图形

LabVIEW 强大的显示功能增强了用户界面的表达能力，除了数据的显示，图形化波形显示是 LabVIEW 在虚拟仪器设计中的特点。

波形显示不单单是几条曲线的显示，根据不同的功能数据输出，可以将波形分为多种，如图 2.11 所示。

下面主要介绍波形图、波形图表、XY 图、强度图和强度图表。

### 1. 波形图

波形图用于将测量值显示为一条或者多条曲线。波形图仅绘制单值函数，即在  $y=f(x)$  中，各点沿  $x$  轴均匀分布。波形图可显示包含任意个数据点的曲线。波形图接收多种数据类型，从而最大限度地降低了数据在显示为图形前进行类型转换的工作量。波形图显示波形是

以成批数据一次刷新方式进行的，数据输入基本形式是数据数组（一维或二维数组）、簇或波形数据。

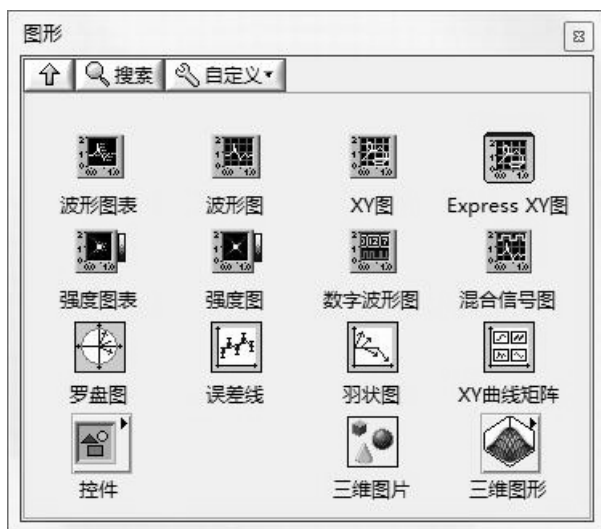


图 2.11 图形显示

波形图是一次性完成显示图形刷新出来的，所以其输入数据必须是完成一次显示所需要的所有数据数组，而不能把测量结果逐次输入，因此不能把随机数函数的输出节点直接与波形图的端口相连。

波形图显示的每条波形的数据都必须是一个一维数组，这是波形图的特点，所以要显示  $n$  条波形就必须有  $n$  组数据。这些数据数组如何组织，用户可以根据不同需要来确定。

应当注意的是，如果不同曲线间的数据量或数据的大小差距太大，则不适合用一个波形图来进行显示。因为波形图总是要在一个显示屏的范围内把一个数组的数据完全显示出来。如果一维数组与另一组数据的数据量相差太大，长度长的波形将被压缩，影响显示效果。

除了数组和簇，波形图还可以显示波形数据。波形数据是 LabVIEW 的一种数据类型，本质上还是簇。

## 2. 波形图表

波形图表是一种特殊的指示器，在图形子选板中找到，选中后拖入前面板即可。

波形图表在交互式数据显示中有 3 种刷新模式：示波器图表、带状图表和扫描图，用户可以在右键菜单的高级中选择刷新模式即可。

示波器图表、带状图表和扫描图在处理数据时略有不同。带状图表有一个滚动显示屏，当新的数据到达时，整个曲线会向左移动，最原始的数据点将移出视野，而最新的数据将会添加到曲线的最右端，这一过程与实验中常见的纸带记录仪的运行方式非常相似。

示波器图表、扫描图表和示波器的工作方式十分相似。当数据点多到足以使曲线到达示波器图表绘图区域的右边界时，将清除整个曲线，并从绘图区的左侧开始重新绘制，扫描图表和示波器图表非常类似，不同之处在于当曲线到达绘图区域的右边界时，不是将旧曲线消除，而是用一条移动的红线标记新曲线的开始，并随着数据的不断增加在绘图区中逐渐移

动。示波器图表和扫描图表比带状图表运行速度快。

波形图表和波形图的不同之处在于：波形图表保存了旧的数据，所保存旧数据的长度可以自行指定。新传给波形图表的数据被接续在旧数据的后面，这样就可以在保持一部分旧数据显示的同时显示新的数据。也可以把波形图表的这种工作方式想象为先进先出的队列，新数据到来之后，会把同样长度的旧数据从队列中挤出去。

### 3. XY 图

波形图和波形图表只能用于显示一维数组中的数据或是一系列单点数据，对于需要显示横、纵坐标的数据，它们就无能为力了。前面讲述的波形图的  $Y$  值对应实际的测量数据， $X$  值对应测量点的序号，适合显示等间隔数据序列的变化。例如，按照一定采样时间采集数据的变化，但是它不适合描述  $Y$  值随  $X$  值变化的曲线，也不适合绘制两个相互依赖的变量（如  $Y/X$ ）。对于这种曲线，LabVIEW 专门设计了 XY 图。

与波形图相同，XY 波形图也是一次性完成波形显示刷新，不同的是 XY 图的输入数据类型是由两组数据打包构成的簇，簇的每一对数据都对应一个显示数据点的  $X$ 、 $Y$  坐标。

### 4. 强度图

强度图是 LabVIEW 提供的另一种波形显示，它用一个二维强度图表示一个三维的数据类型，一个典型的强度图如图 2.12 所示。

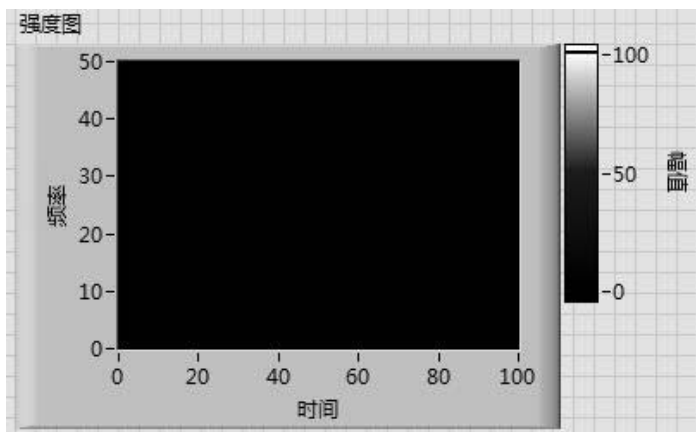


图 2.12 强度图

从图中可以看出强度图与前面介绍过的曲线显示工具在外形上的最大区别是，强度图拥有的标签为幅值的颜色控制组件，如果把标签为时间和频率的坐标轴分别理解为  $X$  轴和  $Y$  轴，则幅值组件相当于  $Z$  轴的刻度。

在使用强度图前先介绍一下颜色梯度，颜色梯度在控制选板的“经典—经典数值”子选板中，当把这个控件放在前面板时，默认建立一个指示器，如图 2.13 所示。

可以看到颜色梯度指示器的左边有一个颜色条，颜色条上有数字刻度，当指示器得到数据输入时，输入值作为刻度在颜色条上对应的颜色显示在控件右侧的颜色框中。若输入值不在颜色条边上的刻度值范围内，当超过 100 时，显示颜色条上方小矩形内的颜色，默认为白

色；当超过下界时，显示颜色条下方小矩形内的颜色，默认为红色。当输入为 100 和 -1 时，分别显示为白色和红色。在编辑和运行程序时，用户可单击上下两个小矩形，这时会弹出颜色拾取器，在里面可定义超界颜色，如图 2.14 所示。

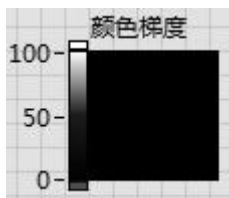


图 2.13 颜色梯度



图 2.14 定义超界颜色

实际上，颜色梯度只包含 5 个颜色值：0 对应黑色，50 对应蓝色，100 对应白色。0~50 之间和 50~100 之间的颜色都是插值的结果。在颜色条上弹出的快捷菜单中选择添加刻度可以增加新的刻度，如图 2.15 所示。添加刻度之后，可以改变新刻度对应的颜色，这样就为刻度梯度增加了一个数值颜色对。

在使用强度图时，要注意其排列顺序，如图 2.16 所示。原数组的第 0 行在强度图中对应于最左边的一列，而且元素对应色块按从下到上排列。值为 100 时，对应的白色在左上方，值为 0 时，对应的黑色在底端的中间。

### 5. 强度图表

与强度图一样，强度图表也是用一个二维的显示结构来表达一个三维的数据类型，它们之间的主要区别在于图像的刷新方式不同：当强度图接收到新的数据时，会自动清除旧数据的显示；而强度图表会把新数据的显示接续到旧数据的后面，这就是波形图和波形图表的区别。

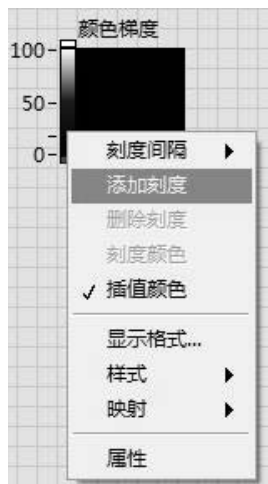


图 2.15 添加刻度

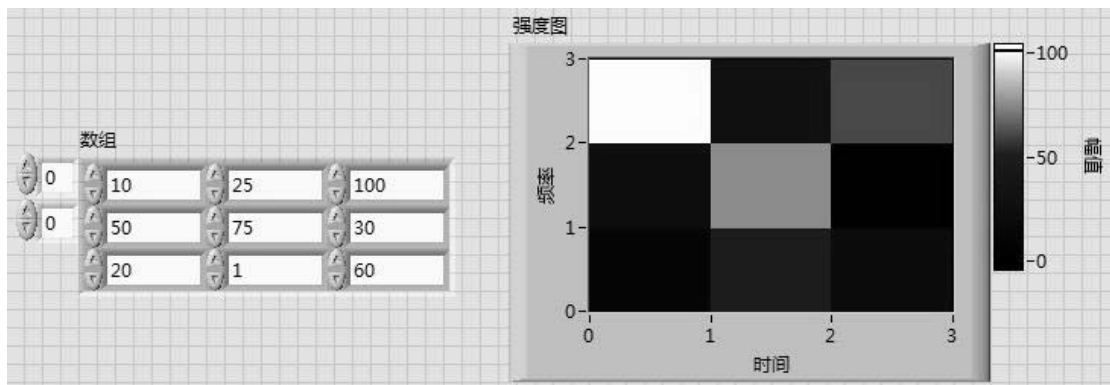


图 2.16 原数组在强度图中的排序

强度图的数据格式为一个二维的数组，它可以一次性地把这些数据显示出来。虽然强度

图表也是接收和显示一个二维的数组数据，但是它显示的方式不一样。它可以一次性显示一列或几列图像，它在屏幕及缓冲区保存一部分旧的图像和数据，每次接收到新的数据时，新的图像紧接着在原有图像的后面显示。当下一列图像将超出显示区域时，会有一列或几列旧图像移出屏幕。数据缓冲区同波形图表一样，也是先进先出，大小可以自己定义，但结构与波形图表（二维）不一样，而强度图表的缓冲区结构是一维的。这个缓冲区的大小是可以设定的，默认为 128 个数据点，若想要改变缓冲区的大小，可以在强度图表上单击右键，从弹出的快捷菜单中选择图表历史长度，即可改变缓冲区的大小，如图 2.17 所示。

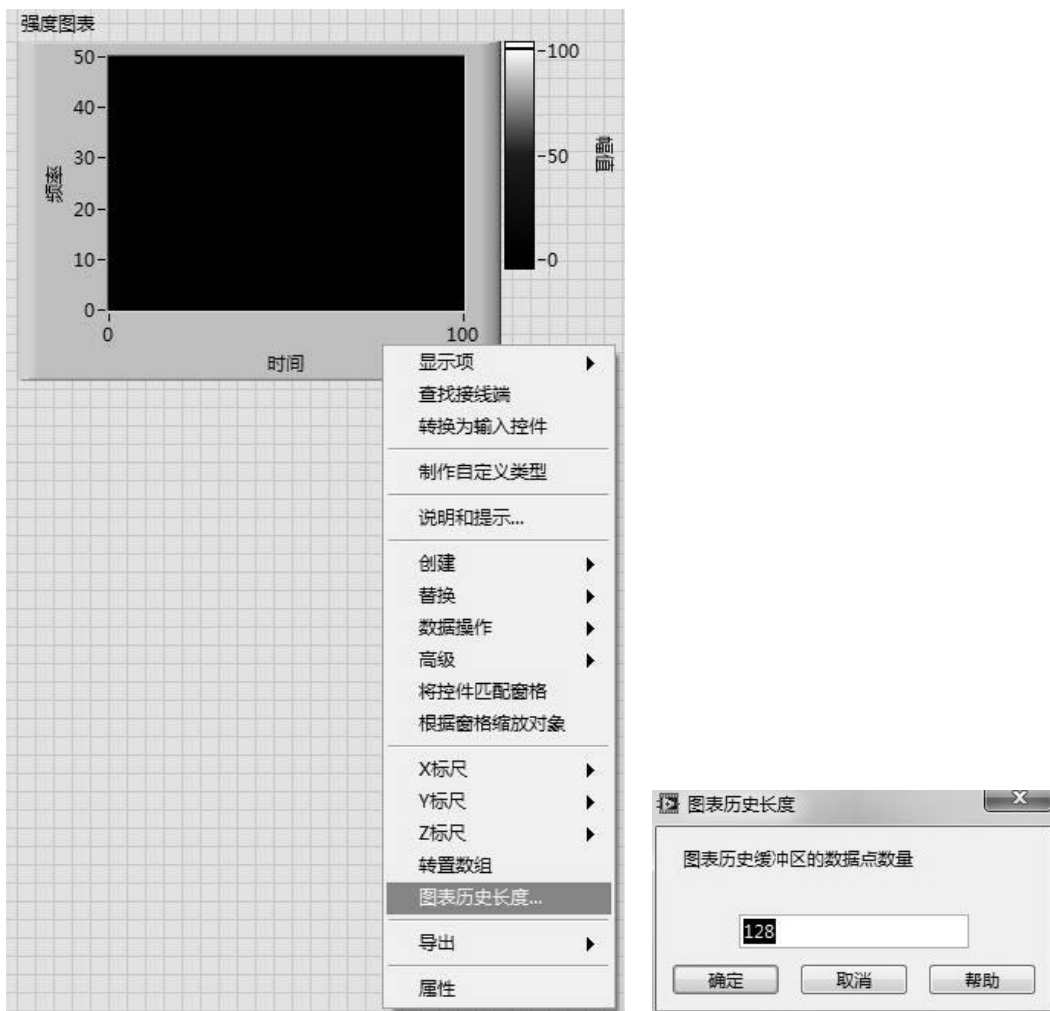


图 2.17 设置图表历史长度

### 2.1.5 三维图形

在很多情况下，把数据绘制在三维空间会更形象和更有表现力。大量实际应用中的数据（例如某个平面的温度分布、联合时频分析、飞机的运动等），都需要在三维空间中可视化显示数据。三维图形可令三维数据可视化，修改三维图形属性可改变数据的显示方式。

LabVIEW 中包含以下三维图形，如图 2.18 所示。



图 2.18 三维图形

- 散点图：显示两组数据的统计趋势和关系。
- 杆图：显示冲激响应并按分布组织数据。
- 彗星图：创建数据点周围有圆圈环绕的动画图。
- 曲面图：在相互连接的曲面上绘制数据。
- 等高线图：绘制等高线图。
- 网格图：绘制有开放空间的网格曲面。
- 瀑布图：绘制数据曲面和 y 轴上低于数据点的区域。
- 箭头图：生成速度曲线。
- 带状图：生成平行线组成的带状图。
- 条形图：生成垂直条带组成的条形图。
- 饼图：生成饼状图。
- 三维曲面图：在三维空间中绘制一个曲面图。
- 三维参数图：在三维空间中绘制一个参数图。
- 三维线条图：在三维空间中绘制线条图。
- ActiveX 三维曲面图：使用 ActiveX 技术，在三维空间中绘制一个曲面图。
- ActiveX 三维参数图：使用 ActiveX 技术，在三维空间中绘制一个参数图。
- ActiveX 三维曲线图：使用 ActiveX 技术，在三维空间中绘制一个曲线图。

前 14 项位于“控件”选板下的“新式—图形—三维图形”子选板中，即图 2.18 (a) 所示；后三项位于“经典—经典图形”子选板下，即图 2.18 (b) 第 4 行所示。

ActiveX 三维图形控件仅在 Windows 平台的 LabVIEW 完整版和专业版开发系统上可用。与其他 LabVIEW 控件不同，这三个三维图形模块不是独立的，实际上它们都是包含了 ActiveX 控件的 ActiveX 容器与某个三维绘图函数的组合。

## 1. 三维曲面图

三维曲面图用于显示三维空间的一个曲面。在前面板放置一个三维曲面图时，程序框图将出现两个图标，如图 2.19 所示。

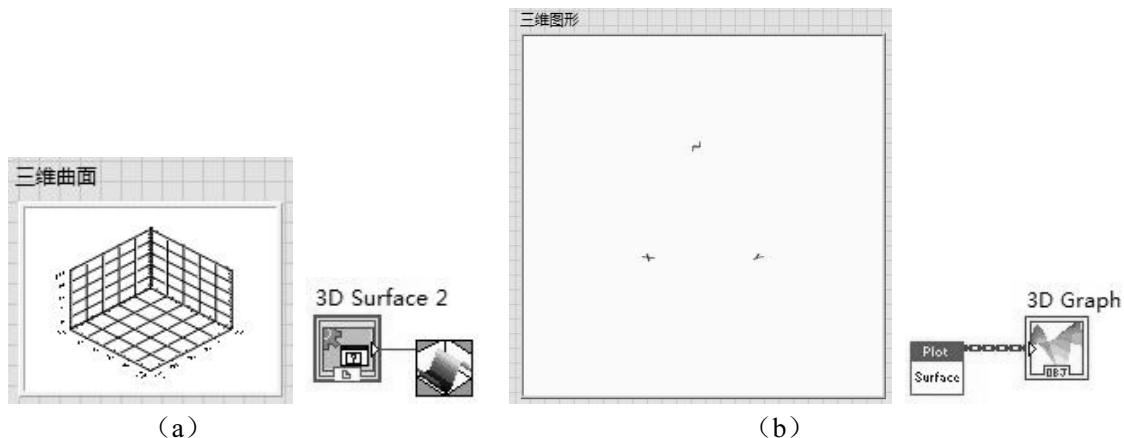


图 2.19 经典选板中的 ActiveX 三维曲面图和新式选板中的三维曲面图

从图 (a) 中可以看出，三维曲面图相应的程序框图由两部分组成：3D Surface 和三维曲面，其中 3D Surface 只负责图形显示，作图则由三维曲面来完成。

三维曲面的图标和端口如图 2.20 所示。三维图形输入端口是 ActiveX 控件输入端，该端口的下面是两个一维数组输入端，用于输入  $X$ 、 $Y$  坐标值。 $Z$  矩阵端口的数据类型为二维数组，用于输入  $Z$  坐标。三维曲面在作图时采用的是描点法，即根据输入的  $X$ 、 $Y$ 、 $Z$  坐标在三维空间中确定一系列数据点，然后通过插值得到曲面。在作图时，三维曲面根据  $X$  和  $Y$  的坐标数组在  $XY$  平面上确定一个矩形网格，每个网格节点都对应三维曲面上的一个点在  $XY$  坐标平面上的投影。 $Z$  矩阵数组给出了每个网格节点所对应的曲面点的  $Z$  坐标，三维曲面根据这些信息就能够完成作图。三维曲面不能显示三维空间的封闭图形，要显示封闭图形应使用三维参数曲面。

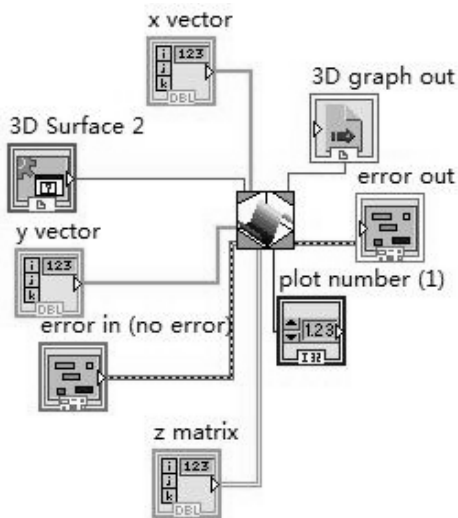


图 2.20 三维曲面的图标和端口



对于前面板的三维曲面图，按鼠标左键并移动鼠标可以改变视点位置，三维曲面图发生了旋转，松开鼠标后将显示新视点的观察图形。

在 LabVIEW 中可以更改三维曲面图形的显示方式，方法是在三维曲面图上单击右键，从弹出的快捷菜单中选择“CWGraph3D”菜单，从下一级菜单中选择“属性”菜单，如图 2.21 所示。



图 2.21 三维曲面图属性的选择

系统弹出属性设置的对话框，同时会出现一个小的 CWGraph3D 控件面板，如图 2.22 所示。



图 2.22 CWGraph3D 控件的属性设置对话框

属性对话框中共有 7 个选项卡，包括 Graph、Plots、Axes、Value Pairs、Format、Cursors 和 About。下面对常用的几项进行介绍，其他各项属性的设置方法相似。

Graph 选项卡中包含 4 部分：General、3D、Light 和 Grid Planes，以及常规属性设置、

三维显示设置、灯光设置和网格平面设置。

**General** 常规属性设置用来设置 CWGraph3D 控件的标题，其中，Font 用于设置标题的字体。Graph frame Visible 用于设置图像边框的可见性。Enable dithering 用于设置是否开启抖动，开启抖动可以使颜色过渡更为平滑。Use 3D acceleration 用于设置是否使用 3D 加速。Caption color 用于设置标题颜色。Background color 用于设置标题的背景颜色。Track mode 用于设置跟踪的时间类型。

**3D** 三维显示设置中的 Projection 用于设置投影类型，有正交投影（Orthographic）和透视（Perspective）。Fast Draw for Pan/Zoom/Rotate 用于设置是否开启快速画法，此项开启时，在进行移动、缩放、旋转时只用数据点来代替曲面，以提高作图速度，默认为选中状态。Clip Data to Axes Ranges 用于设置是否剪切数据，当选中此项时只显示坐标轴范围内的数据，默认为选中状态。View Direction 用来设置视角。User Defined View Direction 用于设置用户视角，共有三个参数：纬度（Latitude）、经度（Longitude）、视点距离（Distance），如图 2.23 所示。

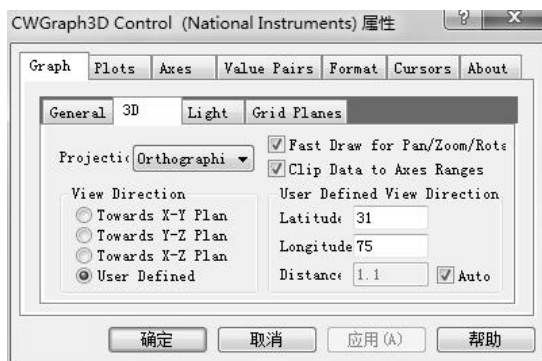


图 2.23 三维显示设置对话框

在 **Light** 灯光选项卡中处理默认的光照，CWGraph3D 控件还提供了 4 个可控制的灯。Enable Lighting 用于设置是否开启辅助灯光照明。Ambient Color 用于设置环境光的颜色。

Enable Light 用于设置具体每盏灯的属性，包括纬度（Latitude）、经度（Longitude）、视点距离（Distance）和衰减（Attenuation），如图 2.24 所示。

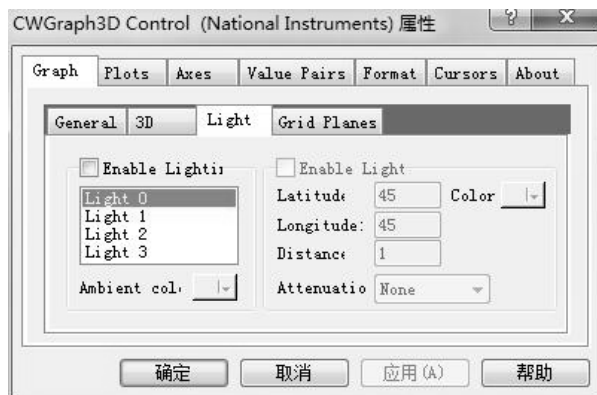


图 2.24 灯光设置对话框

在 Grid Plane 网格平面设置中, Show Grid Plane 用于设定显示网格的平面, Smooth grid lines 用来选中该项以平滑网格线。Grid frame color 用于设置网格边框颜色, 如图 2.25 所示。



图 2.25 网格平面设置对话框

在 CWGraph3D 的 Plots 选项卡中, 可以更改图形的显示风格。Plots 选项卡对话框如图 2.26 所示。



图 2.26 Plot 选项卡对话框

若要改变显示风格, 可单击 Plot Style 按钮, 将显示 9 种风格, 如图 2.27 所示。

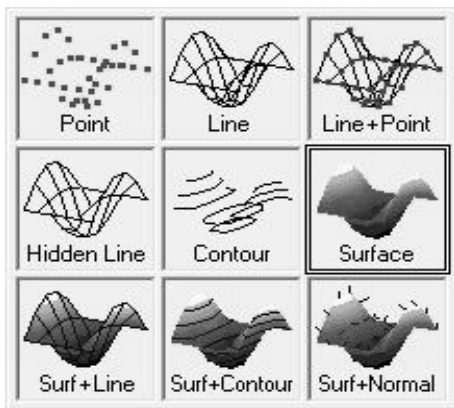


图 2.27 图形的显示风格

在三维曲面图中，经常会用到光标，用户可以在 CWGraph3D 的 Cursors 选项卡中选择。添加方法是单击 Add 按钮，设置需要的坐标即可，如图 2.28 所示。添加了光标的三维曲面图如图 2.29 所示。

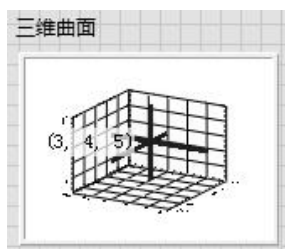
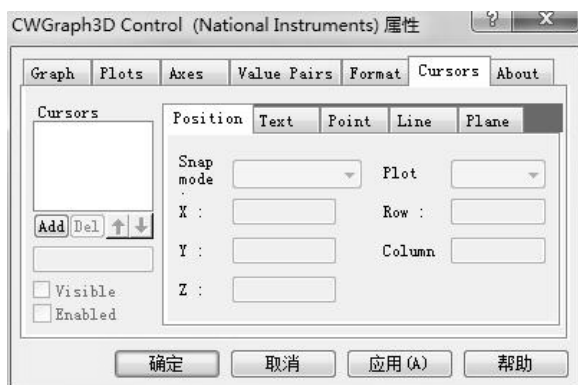


图 2.28 光标添加对话框 图 2.29 添加了光标的三维曲面图

## 2. 三维参数图

三维曲面可以显示三维空间的一个曲面，但是在显示三维空间的封闭图形时就无能为力了，这时需要使用三维参数图，如图 2.30 所示是三维参数图的前面板显示和程序框图。在其程序框图中将出现两个图标：一个是 3D parametric Surface，另一个是三维参数曲面。

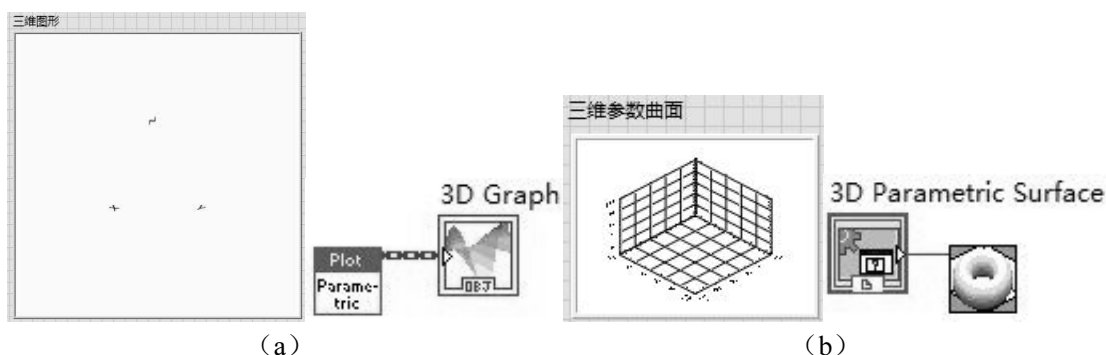


图 2.30 新式选板中的三维参数图和经典选板中的 ActiveX 三维参数图

图 2.31 所示为三维参数曲面，三维参数曲面各端口的含义：三维图形表示 3D Parametric 输入端， $\mathbf{x}$  矩阵表示参数变化时  $x$  坐标所形成的二维数组； $\mathbf{y}$  矩阵表示参数变化时  $y$  坐标所形成的二维数组； $\mathbf{z}$  矩阵表示参数变化时  $z$  坐标所形成的二维数组。三维参数曲面的使用较为复杂，但借助参数方程的形式可以很容易理解，需要 3 个方程： $x=f_x(i, j)$ ； $y=f_y(i, j)$ ； $z=f_z(i, j)$ 。其中， $x$ 、 $y$ 、 $z$  是图形中点的三维坐标， $i$ 、 $j$  是两个参数。

## 3. 三维曲线图

三维曲线图是用于显示三维空间的一条曲线。三维曲线图的新式选板中的三维曲线图和经典选板中的 ActiveX 三维曲线图如图 2.32 所示。程序框图中将出现两个图标。一个是 3D

Curve 图标，另一个是三维曲线的图标。

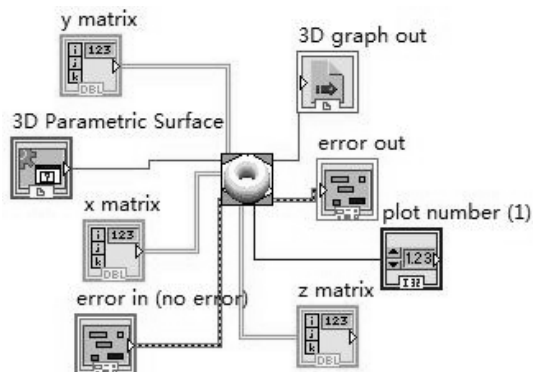


图 2.31 三维参数曲面的图标和端口

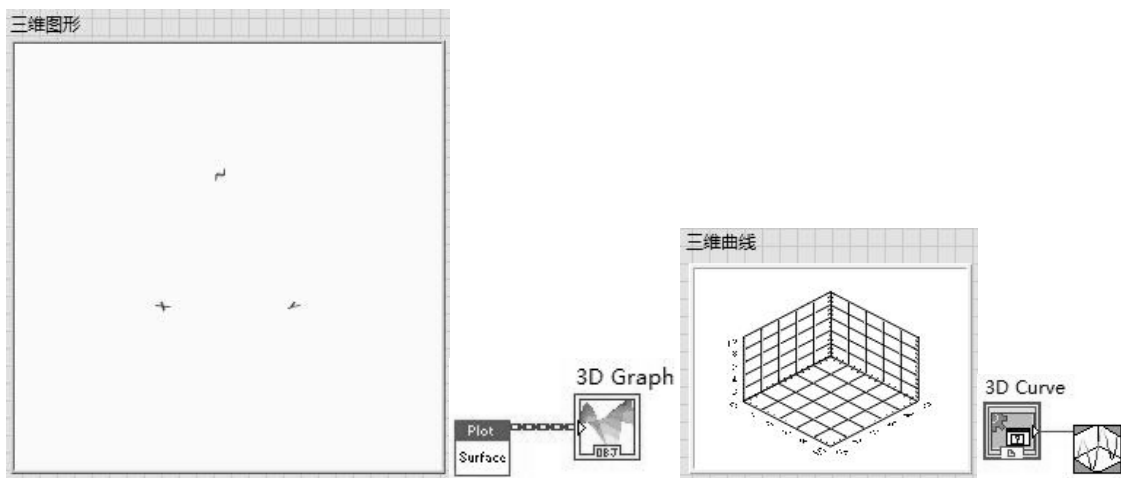


图 2.32 新式选板中的三维曲线图和经典选板中的 ActiveX 三维曲线图

如图 2.33 所示，三维曲线有 3 个重要的输入数据端口，分别是  $x$  向量、 $y$  向量和  $z$  向量，对应曲线的 3 个坐标向量。在编写程序时，只要分别在 3 个坐标向量上连接一维数组数据就可以显示三维数组。

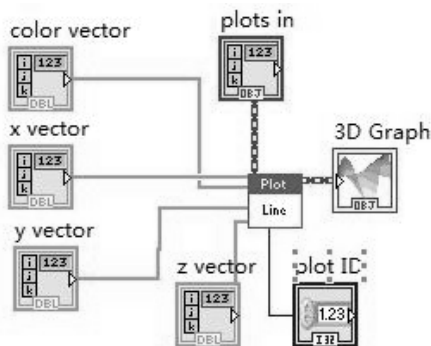


图 2.33 三维曲线的图标及其端口

三维曲线图有属性浏览器窗口，通过属性浏览器窗口用户可以很方便地浏览并修改对象的属性，在三维曲线图上单击鼠标右键，从弹出的快捷菜单中选择“属性浏览器”，将弹出三维曲线“属性浏览器”窗口，如图 2.34 所示。

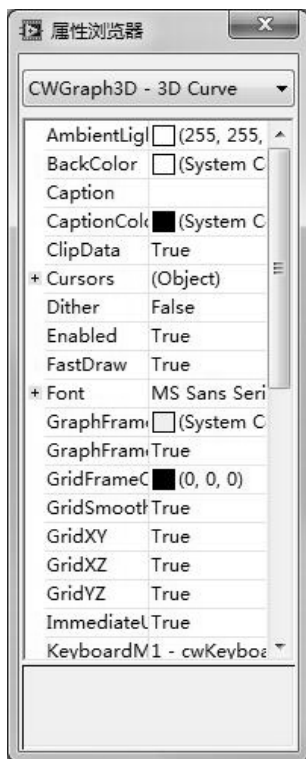


图 2.34 “属性浏览器”窗口

## 2.1.6 极坐标图

极坐标图实际上就是一个图片控件，极坐标图的使用相对简单，极坐标图在新式选板的“图形→控件”子选板和经典选板的“图形→控件”子选板中，如图 2.35 所示。



图 2.35 “控件”选板

极坐标图的前面板和程序框图如图 2.36 所示。

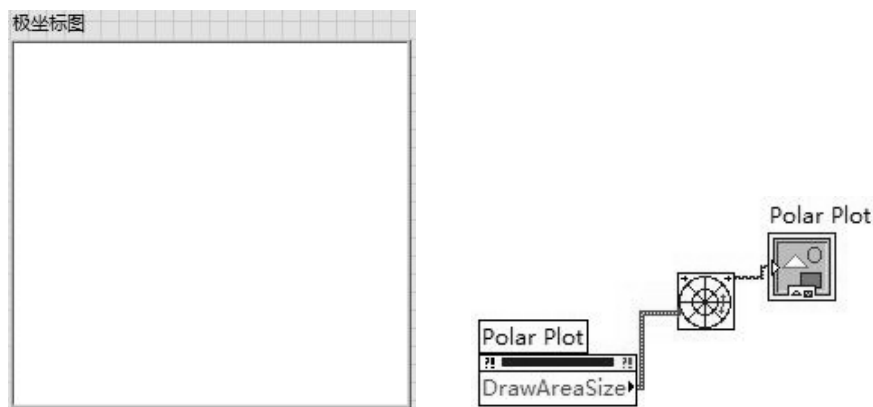


图 2.36 极坐标图的前面板和程序框图

在使用极坐标图时，需要提供以极径、极角方式表示的数据点的坐标。极坐标图的图标和端口如图 2.37 所示。数据数组[大小、相位（度）]端口连接点列的坐标数组，尺寸（宽度、高度）端口设置极坐标图的尺寸。在默认设置下，该尺寸等于新图的尺寸。极坐标属性端口用于设置极坐标图的图形颜色、网格颜色和显示象限等属性。

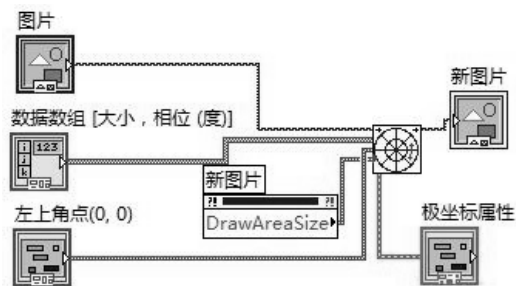


图 2.37 极坐标图的图标和端口

## 2.2 程序框图

框图是程序代码的图形化表示，可通过选择主菜单项中“窗口→显示程序框图”或者使用快捷键 **Ctrl+E** 在前面板和框图之间切换。

框图中含有各种图形化的函数、子 VI、变量、常量、结构和连线，如图 2.38 所示，从前面板中得到用户输入的数据后，按照所编写的程序以数据流方式进行运算，最后给出结果并反馈给用户。

图标是 VI 的图形符号，LabVIEW 中内置了图标编辑器，供用户自行绘制个性化图标，用户可以通过在界面右上角的 VI 图标上直接双击鼠标调出图标编辑器。

由框图组成的图形对象共同构造出通常所示的源代码。框图与文本编辑语言中的文本行相对应。事实上，框图是实际可执行的代码。框图是通过将完成特定功能的对象连接在一起构建出来的。



图 2.38 图形化的编程元素

程序框图由以下 3 种组件构建而成，如图 2.39 所示。

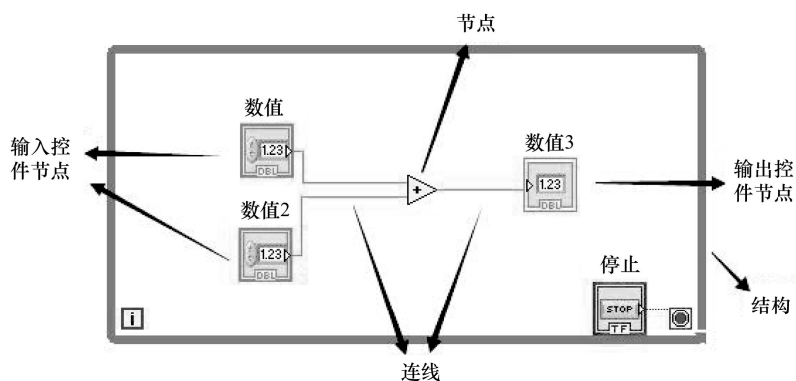


图 2.39 框图的组成





节点：是程序框图上的对象，具有输入、输出端，在 VI 运行时进行运算。节点相当于文本编辑语言中的语句、运算符、函数和子程序。

接线端：用于表示输入控件或显示控件的数据类型。在程序框图中可以将前面板的输入控件或显示控件显示为图标或数据类型接线端。在默认状态下，前面板对象显示为图标接线端。

连线：程序框图中对象的数据传输通过连线实现。每根连线都只有一个数据源，但可以与多个读取该数据的 VI 和函数连接。不同数据类型的连线有不同的颜色、粗细和样式。断开的连线显示为黑色的虚线，中间有个红色的“×”。出现断线的原因有很多，例如，当试图连接两个数据类型不兼容的对象时就会产生断线。

LabVIEW 有以下类型的节点，如表 2.1 所示。

表 2.1 LabVIEW 的各类型节点及其功能

名 称	功 能
函数	内置的执行元素，相当于操作符、函数或语句
子 VI	用于另一个 VI 程序框图上的 VI，相当于子程序
Express VI	协助常规测量任务的子 VI。Express VI 是在配置对话框中配置的
结构	执行控制元素，如 For 循环、While 循环、条件结构、平铺式和层叠式顺序结构、定时结构和事件结构
公式节点和表达式节点	公式节点可以直接向程序框图输入方程，其大小可以调节。表达式节点用于计算含有单变量的表达式或方程
属性节点和调用节点	属性节点用于设置或寻找类的属性。调用节点用于设置对象执行方式
通过引用节点调用	用于调用动态加载的 VI
调用库函数节点	调用大多数标准库或 DLL
代码接口节点（CIN）	调用以文本编辑语言所编写的代码

## 2.3 连 线 板

连线板用来定义输入和输出，用户可在界面右上角的 VI 图标上用鼠标右键单击，选择“显示连线板”，调出当前 VI 的连线板进行设置。

在 LabVIEW 编程中，用户要做的工作就是恰当地设置参数，并连接各个子 VI。编程一般步骤就是使用鼠标选取合适的模块、连线和设置参数的过程，这与烦琐枯燥的文本编程相比更为简单、生动和直观。

如果将虚拟仪器与传统仪器类比，前面板就像是仪器的操作和显示面板，提供各种参数和数据的显示，框图就像是仪器内部的印制电路板，是仪器的核心运算部分，对仪器用户来讲则是透明的，而图标和连线板可以比作电路板上的电子元器件和集成电路，保证了仪器正常的逻辑和运算功能。

## 2.4 对象的选择与删除

新建 VI 后，还需要对 VI 进行编辑，使 VI 的图形化交互式用户界面更加美观、友好而易于操作，使 VI 框图程序的布局 and 结构更加合理，易于理解、修改。

### 1) 选择对象

在工具选板中将鼠标切换为对象操作工具。当选择单个对象时，直接用鼠标左键单击需要选中的对象，或者按住<Shift>键，用鼠标左键单击多个目标对象。

### 2) 删除对象

选中对象按<Delete>键，或在窗口菜单栏中选择“编辑→删除”命令，即可删除对象。

### 3) 变更对象位置

使用对象操作工具拖动目标对象到指定位置，在拖动对象时，窗口会出现一个红色的文本框，实时显示对象移动的相对坐标。

## 2.5 设置对象的位置关系

在 LabVIEW 程序中，设置多个对象的相对位置关系是修饰前面板过程中一件非常重要的工作。LabVIEW 2013 提供了专门用于调整多个对象位置关系以及设置对象大小的工具，它们位于 LabVIEW 的工具栏上。

### 2.5.1 对齐关系

LabVIEW 所提供的用于修改多个对象位置关系的工具如图 2.40 所示。这几种工具分别用于调整多个对象的对齐关系以及调整对象之间的距离。

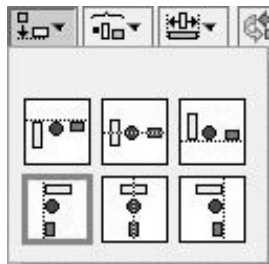


图 2.40 对齐对象工具

选中需要对齐的对象，然后在工具条中单击“对齐对象”按钮，会出现一个图形化的下

拉菜单。在下拉菜单中可以选择各种对齐方式。选单中的各种图标很直观地表示了各种不同的对齐方式，有左边缘对齐、右边缘对齐、上边缘对齐、下边缘对齐、水平中轴线对齐以及垂直中轴线对齐 6 种方式可选。

要将几个对象按左边缘对齐，步骤如下。

- (1) 选中目标对象。
- (2) 在“对齐对象”下拉菜单中选择“左边缘”对齐。

## 2.5.2 分布对象

选中对象，在工具条中单击“分布对象”按钮，会出现一个图形化的下拉菜单，如图 2.41 所示。在菜单中可以选择各种分布方式。选单中的各图标很直观地表示了各种不同的分布方式。

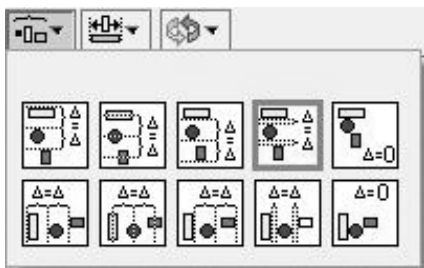


图 2.41 分布对象

例如，要将对象按照等间隔垂直分布步骤如下。

- (1) 选中目标对象。
- (2) 在分布对象下拉菜单中选择“垂直间隔”。

## 2.5.3 改变对象在窗口中的前后次序

选中对象，在工具栏中单击“重新排序”按钮，可以在下拉菜单中改变对象在窗口中的前后次序，下拉菜单如图 2.42 所示。



图 2.42 “重新排序”下拉菜单

“向前移动”是将对象向上移动一层；“向后移动”是将对象向下移动一层；“移至前面”是将对象移至窗口的最顶层；“移至后面”是将对象移动至窗口的最底层。

例如，要将一个对象从窗口的最顶层移动至窗口的最底层，具体操作步骤如下。

(1) 选中目标对象。

(2) 在“重新排序”下拉菜单中选择“移至后面”。

## 2.5.4 组合与锁定对象

在“重新排序”下拉菜单中还有几个选项，它们分别是“组合”和“取消组合”“锁定”和“解锁”。

“组合”的功能是将几个选定的对象合成一个对象组，对象组中的所有对象形成一个整体，它们的相对位置和相对尺寸都相对固定。当移动对象组或改变对象组的尺寸时，对象组中所有的对象同时移动相同的距离或改变相同的尺寸。注意，“组合”的功能仅仅是将数个对象按照其位置和尺寸简单地组合在一起形成一个整体，并没有在逻辑上将其组合，它们之间在逻辑上的关系并没有因为组合在一起而得到改变。“取消组合”的功能是解除对象组中对象的组合，将其还原为独立的对象。

“锁定”的功能是将几个选定的对象组合成一个对象组，并且锁定该对象组的位置和大小，用户不能改变锁定对象的位置和尺寸。当然，用户也不能删除处于锁定状态的对象。“取消锁定”的功能是解除对象的锁定状态。

当用户已经编辑好一个 VI 的前面板时，建议用户利用“组合”或者“锁定”功能将前面板中的对象组合并锁定，防止由于误操作而改变了前面板对象的布局。

## 2.5.5 网格布局

网格可以作为排列控件的参考，显示与隐藏可选择菜单栏中的“工具→选项”命令，弹出“选项”对话框，选择“前面板”选项。

在“前面板网格”选项下设置前面板网格，包括“显示前面板网格”、“默认前面板网格大小”、“前面板背景对比度”、“启用前面板网格对齐”、“缩放新对象以匹配网格大小”和“对齐网格绘制样式”。

# 第 3 章 LabVIEW 的编程环境

## 3.1 启动界面和文件管理

成功安装 LabVIEW 之后，双击桌面上的图标或单击“开始”菜单中的对应项，即可启动 LabVIEW。LabVIEW 2013 的启动界面如图 3.1 所示。



图 3.1 LabVIEW 启动界面

启动界面上的向导框内列出了最常用的命令，如新建 VI、新建项目、新建基于模板的 VI 以及最近打开过的项目和 VI 程序，同时还可以查看各种信息和资源，如用户手册、帮助主题以及 National Instruments 网站上的各种资源等，以方便用户使用。

### 3.1.1 新建 VI

创建 VI 是 LabVIEW 编程应用中的基础，下面详细介绍如何创建 VI。

选择菜单栏中的“新建—新建 VI”命令，弹出如图 3.2 所示的新建 VI 窗口。前面是 VI 的前面板窗口，后面是 VI 的程序框图窗口，在两个窗口的右上角是默认的 VI 图标/连线板。

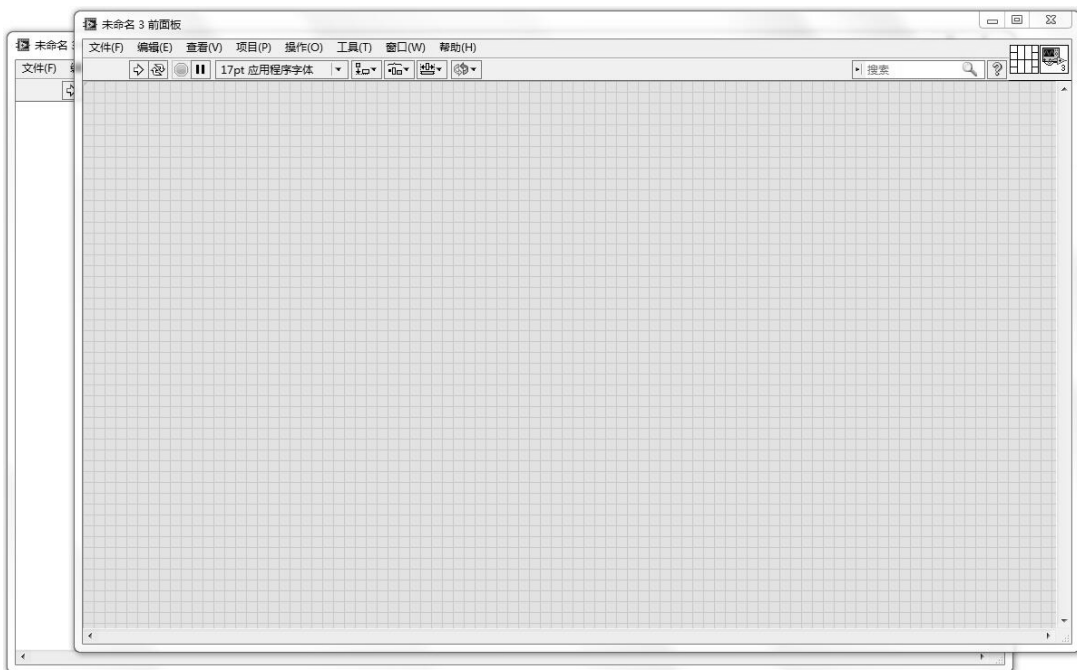


图 3.2 新建 VI 窗口

### 3.1.2 编辑 VI 图标

双击前面板窗口或框图程序窗口右上角的 VI 图标，或在 VI 图标处单击鼠标右键，并在弹出的快捷菜单中选择“编辑图标”，将弹出“图标编辑器”对话框，如图 3.3 所示。

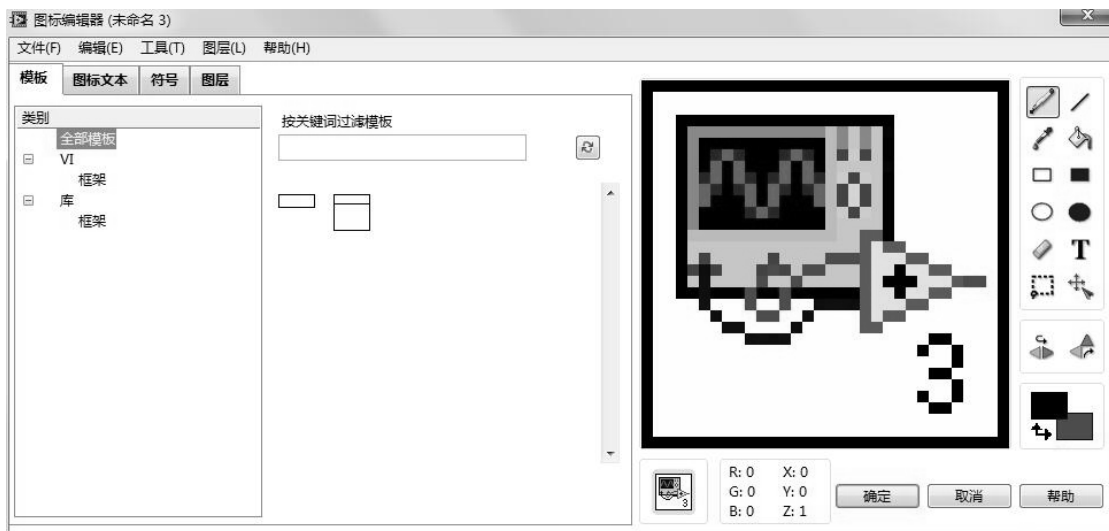


图 3.3 “图标编辑器”对话框

该对话框包括的部分如表 3.1 所示。

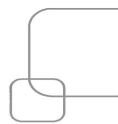


表 3.1 “图标编辑器”对话框包含的各部分及其功能

名 称	功 能
模板	显示作为图标背景的图标模板。显示 LabVIEW Data 或 Icon Templates 目录中的所有.png、.bmp 和.jpg 文件
图标文本	指定在图标中显示的文本
符号	显示图标中可包含的符号。“图标编辑器”对话框可显示 LabVIEW Data 或 Glyphs 中所有的.png、.bmp 和.jpg 文件。在默认情况下, 该页包含 NI 公司网站上图标库中所有的符号。选择“工具—同步 ni.com”图标库, 打开“同步图标库”对话框, 使 LabVIEW Data 或 Glyphs 目录最新的图标库保持同步
图层	显示图标图层的所有图层。如未显示该页, 选择“图层—显示所有图层”可显示该页
图标	显示图标的实际大小预览。图标显示可通过“图标编辑器”对话框进行更改
预览	显示图标的放大预览。预览可显示通过“图标编辑器”对话框进行更改
RGB	显示光标所在位置像素的 RGB 颜色组成
XYZ	显示光标所在位置像素的 X-Y 位置。Z 值为图标的用户图层总数
工具	用于显示手动修改图标的编辑工具。如使用编辑工具时单击左键, LabVIEW 将使用线条颜色工具; 使用编辑工具时单击右键, LabVIEW 将使用填充颜色工具

如需创建自定义编辑环境, 可修改“图标编辑器”对话框。在修改“图标编辑器”对话框前, 应保存位于 labview\resource\plugins 的原有文件 1v\_icon.vi 和 IconEditor 文件夹。创建自定义图标编辑器时, 可使用 labview\resource\plugins\IconEditor\Discover Who Invoked the Icon Editor.vi 目录中的“搜索图标库调用方”VI 获取当前编辑项图标的名称、路径和应用程序引用。通过该信息可自定义图标。

### 3.1.3 保存 VI

在前面板窗口或程序框图窗口中选择菜单栏中的“文件→保存”命令, 然后在弹出的保存文件对话框中选择适当的路径和文件名保存该 VI。如果一个 VI 在修改后没有存盘, 那么在 VI 的前面板和程序框图窗口的标题栏中就会出现一个“\*”, 提醒用户注意存盘。

### 3.1.4 新建文件

单击启动界面上的新建 VI 图标, 可以建立一个空白的 VI。

单击启动界面中文件菜单下的“新建”按钮, 将打开如图 3.4 所示的“新建”对话框, 在这里可以通过多种方式来建立文件。

利用“新建”对话框, 可以创建 3 种类型的文件, 分别是 VI、项目和其他文件。

其中, 新建 VI 是经常使用的功能, 包括新建空白 VI、创建多台 VI 以及基于模板创建 VI。如果选择 VI, 将创建一个空的 VI, VI 中的所有空间都需要用户自行添加。如果选择基于模板, 有很多种程序模板供用户选择, 如图 3.5 所示。



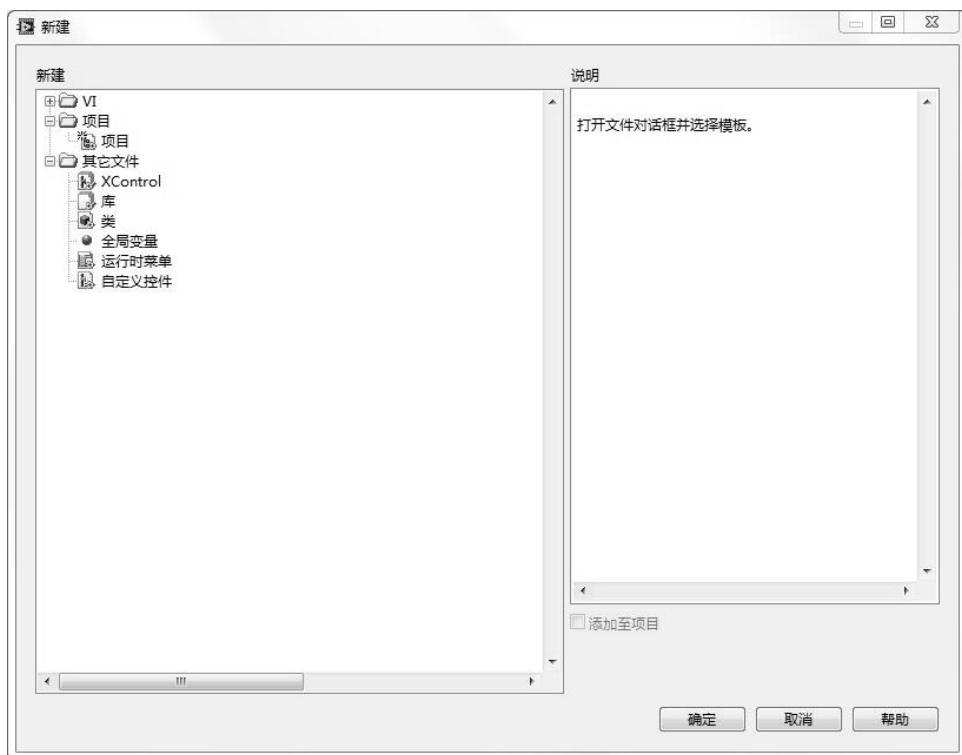


图 3.4 “新建”对话框

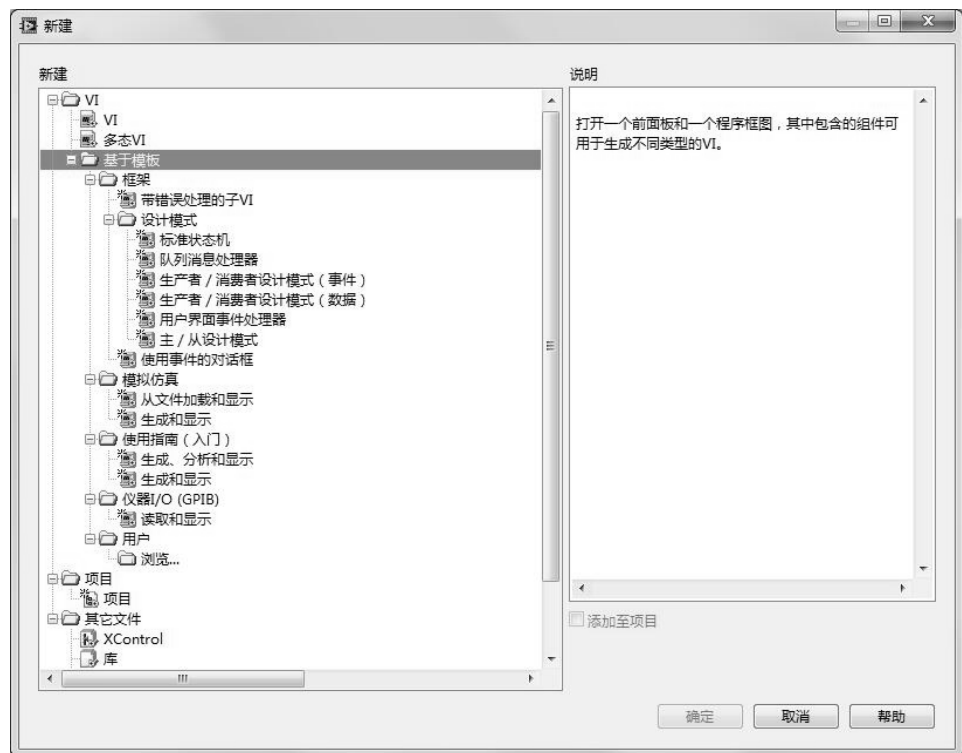


图 3.5 基于模板选项的新建文件



新建项目包括空白项目文件和基于向导的项目。

其他文件则包括库、类、全局变量、运行时菜单和自定义控件。

用户根据需要可以选择相应的模板进行程序设计，在各种模板中，LabVIEW 已经预先设置了一些组件构成了应用程序的框架，用户只需要对程序框图进行一定程度的修改和功能上的增减就可以在模板的基础上构建自己的应用程序。

### 3.1.5 创建项目

在启动界面单击“创建项目”按钮，或选择菜单栏中的“文件→创建项目”命令，弹出“创建项目”对话框，如图 3.6 所示。



图 3.6 “创建项目”对话框

“创建项目”对话框主要分为左右两部分，分别是文件和资源。在这个界面用户可以选择新建空白 VI、新建空的项目、简单状态机，并且可以打开已有的程序。同时，用户也可以从这个界面获得帮助支持，例如，可以查找 LabVIEW 2013 的帮助文件、互联网上的资源以及 LabVIEW 2013 的程序范例等。

在 LabVIEW 2013 的启动界面上有文件、操作、工具和帮助 4 个菜单，在以下几节中将详细介绍 LabVIEW 2013 的操作选板。

## 3.2 控件选板

控件是 LabVIEW 图形语言的基石，没有控件，LabVIEW 编程语言就是一纸空谈，因此，对控件的熟悉和掌握，对读者学习该语言至关重要。

随着 LabVIEW 的不断升级，控件样式越来越多，功能越来越合理，但系统仍能保留旧版控件，因此控件数量直线上升，同时，图形化语言的表达能力也越来越强。

系统控件的外观取决于 VI 运行的平台，因此，在 VI 中创建的控件外观应与所有 LabVIEW 平台兼容。在不同的平台上运行 VI 时，系统控件将改变其颜色和外观，与该平台的标准对话框控件相匹配。

如图 3.7 所示，控件分为 6 类：新式、银色、系统、经典、Express、.NET 与 Active，下面对这 6 类控件做详细介绍。



图 3.7 控件选板

### 3.2.1 新式控件

新式控件包含编程常用的大部分控件，如图 3.7 所示。

#### 1. 数值型控件

数值选板上的数值对象可用于创建滑动杆、滚动条、旋钮、转盘和数值显示框，如图 3.8 所示。



3.8 “数值”选板

## 2. 布尔型控件

布尔型控件可用于创建按钮、开关和指示灯，如图 3.9 所示。



图 3.9 “布尔”选板图

## 3. 字符串和路径控件

字符串和路径控件可用于创建文本输入框和标签、输入或返回文件或目录的地址，如图 3.10 所示。



3.10 “字符串和路径”选板

#### 4. 数组、矩阵与簇控件

数组、矩阵和簇控件可以用来创建数组、矩阵和簇。数组是同一类型数据元素的集合。簇将不同类型的数据元素归为一组。矩阵是若干行列实数或复数数据的集合，用于线性代数等数学操作，如图 3.11 所示。



图 3.11 “数组、矩阵与簇”选板

#### 5. 列表、表格和树形控件

列表框控件用于向用户提供一个可供选择的列表，如图 3.12 所示。



图 3.12 “列表、表格和树”选板

### 1) 列表框控件

列表框可配置为单选或多选。多选列表可显示更多条目信息，如大小和创建日期等。

### 2) 树形控件

树形控件用于向用户提供一个可供选择的层次化列表。用户将输入树形控件的项组织为若干组项或若干组节点。单击节点旁边的展开符号可展开节点，显示节点中的所有项，单击节点旁的符号还可以折叠节点。

### 3) 表格控件

表格控件可用于在前面板上创建表格。

## 6. 图形控件

图形控件可用于以图形和图表的形式绘制数值数据，如图 3.13 所示。

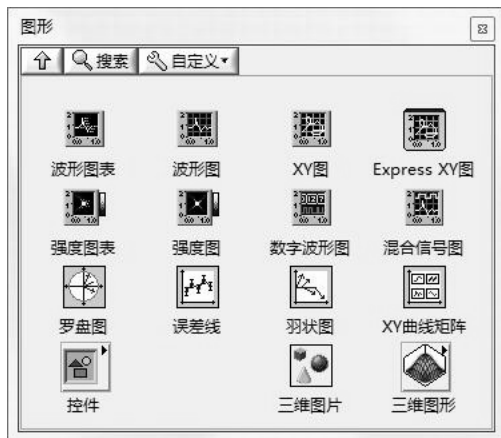


图 3.13 “图形”选板图

关于图形和图表的详细介绍参见本书后面章节。

## 7. 下拉列表与枚举控件

下拉列表与枚举控件用于创建可循环浏览的字符串列表，如图 3.14 所示。



3.14 “下拉列表与枚举”选板

### 1) 下拉列表控件

下拉列表控件是将数值与字符串或图片建立关联的数值对象。下拉列表控件以下拉菜单的形式出现，用户可以在循环浏览的过程中进行选择。下拉列表控件可用于选择互斥项，如触发模式。例如，用户可在下拉列表控件中，从连续、单次和外部触发中选择一种模式。

### 2) 枚举控件

枚举控件用于向用户提供一个可供选择的列表。枚举控件类似于文本或菜单下拉列表控件，但是，枚举控件的数据类型包括控件中所有项的数值和字符串标签的相关信息，下拉列表控件则为数值型控件。

## 8. 容器控件

容器控件可用于组合控件，或在当前 VI 的前面板上显示另一个 VI 的前面板。Windows 容器控件还可以用于在前面板上显示 .NET 和 ActiveX 对象，如图 3.15 所示。



图 3.15 “容器”选板

### 1) 选项卡控件

选项卡控件用于将前面板的输入控件和显示控件重叠放置在一个较小的区域内。选项卡控件由选项卡标签组成，可将前面板对象放置在选项卡控件的每一个选项卡中，并将选项卡标签作为显示不同页的选择器。可使用选项卡控件组合在操作某一阶段需要用到的前面板对象。例如，某 VI 在测试开始前可能要求用户先设置几个选项，然后在测试过程中允许用户修改测试的某些方面，最后允许用户显示和存储相关数据。在程序框图上，选项卡控件默认为枚举控件。选项卡控件中的控件接线端与程序框图上的其他控件接线端在外观上是一致的。

### 2) 子面板控件

子面板控件用于在当前 VI 的前面板上显示另一个 VI 的前面板。例如，子面板控件可用于设计一个类似向导的用户界面。在顶层 VI 的前面板上放置“上一步”和“下一步”按钮，并用子面板控件加载向导中每一步的前面板。

## 9. I/O 控件

I/O 控件可将所配置的 DAQ 通道名称、VISA 资源名称和 IVI 逻辑名称传递至 I/O VI，与仪器或 DAQ 设备进行通信。I/O 名称常量位于函数选板上。常量是在程序框图上向程序框图提供固定值的接线端。如图 3.16 所示是“I/O”选板。



图 3.16 “I/O”选板

### 1) 波形控件

波形控件可用于对波形中的单个数据元素进行操作。波形数据类型包括波形的数据、起始时间和时间间隔。

### 2) 数字波形控件

数字波形控件可用于对数字波形中的单个数据元素进行操作。

### 3) 数字数据控件

数字数据控件显示行列排列的数字数据。数字数据控件可用于创建数字波形或显示从数字波形中提取的数字数据。将数字波形数据输入控件连接至数字数据显示控件，可查看数字波形的采样和信号。

## 10. 变体与类控件

变体与类控件可在前面板中放置变体和 LabVIEW 对象控件，其选板如图 3.17 所示。



图 3.17 “变体与类”选板

## 11. 修饰控件

修饰控件包括一系列线、箭头、方框、圆形、三角形等形状的修饰模块，这些模块如同搭建美观的程序界面的积木，合理组织、搭配这些模块可以构造出绚丽的程序界面。该控件可对前面板对象进行组合或分隔。这些对象仅用于修饰，并不显示数据。

在前面板上放置修饰后，使用重新排序下拉菜单可对层叠的对象重新排序，也可在程序框图上使用修饰，“修饰”选板如图 3.18 所示。

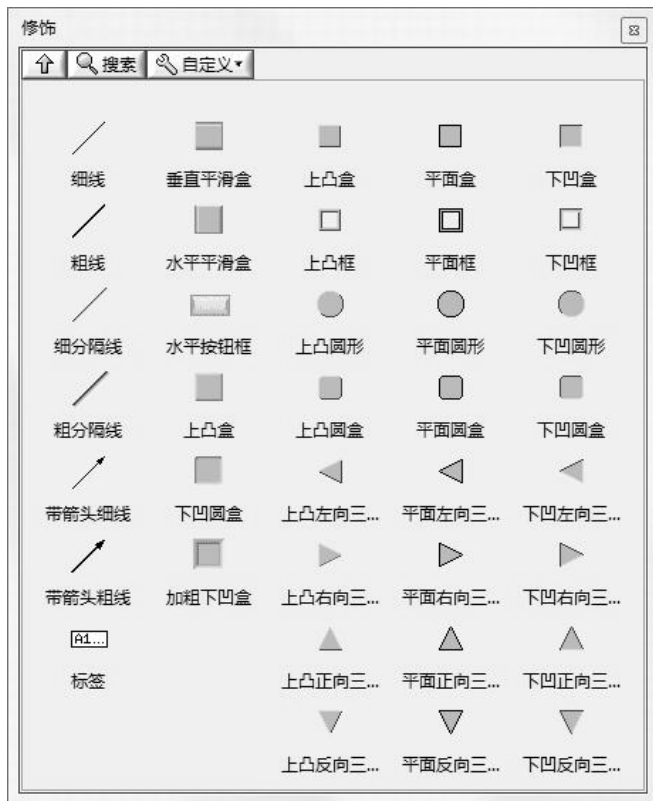


图 3.18 “修饰”选板

“修饰”选板中的各种控件只有其前面板的图形，而没有在程序框图中与之相对应的图标，这些控件的主要功能就是进行界面的修饰，是 LabVIEW 中最为特殊的前面板控件。将这些控件进行适当组合，可以设计出非常美观的程序界面。



## 12. 引用句柄控件

引用句柄控件可用于对文件、目录、设备和网络连接进行操作，将前面板对象信息传送给 VI，其选板如图 3.19 所示。



图 3.19 “引用句柄”选板

引用句柄是对象的唯一标识符，这些对象包括文件、设备或网络连接等。打开一个文件、设备或网络连接时，LabVIEW 会生成一个指向该文件、设备或网络连接的引用句柄。对打开的文件、设备或网络连接进行的所有操作均使用引用句柄来识别每个对象。引用句柄控件用于将一个引用句柄传进或传出 VI，例如，引用句柄控件可在不关闭或不重新打开文件的情况下修改其指向的文件内容。

由于引用句柄是打开对象的一个临时指针，因此它仅在对象打开期间有效。如关闭对象，LabVIEW 会将引用句柄与对象分开，引用句柄即失效。如再次打开对象，LabVIEW 将创建一个与第一个引用句柄不同的新引用句柄。LabVIEW 将为引用句柄所指的對象分配内存空间。关闭引用句柄，该对象就会从内存中释放出来。

由于 LabVIEW 可以记住每个引用句柄所指的信息，如读取或写入对象当前的地址和用户访问情况，因此可以对单一对象执行相互独立的操作。如一个 VI 多次打开同一个对象，那么每次的打开操作都将返回一个不同的引用句柄。VI 结束运行时 LabVIEW 会自动关闭引用句柄，但如果用户在结束使用引用句柄时就将其关闭，可以最有效地利用内存空间和其他资源，这是一个良好的编程习惯。关闭引用句柄的顺序与打开相反。例如，如对象 A 先获得了一个引用句柄，然后在对象 A 上调用方法以获得一个指向对象 B 的引用句柄，在关闭时应先关闭对象 B 的应用句柄，然后再关闭对象 A 的引用句柄。

### 3.2.2 经典控件

许多前面板对象具有高彩色位的外观。为了获取对象的最佳外观，显示器最低应设置为 16 色位。经典选板上的控件时域创建在 256 色和 16 色显示器上显示的 VI。“经典”控件如图 3.20 所示。



图 3.20 “经典”控件

### 1) 经典数值控件

与新式子选板上的控件相比，经典数值选板上还有经典颜色盒和经典颜色梯度选项，用于设置颜色值，如图 3.21 所示。



图 3.21 “经典数值”控件

## 2) 经典布尔控件

经典布尔选板上的“经典布尔”控件如图 3.22 所示。

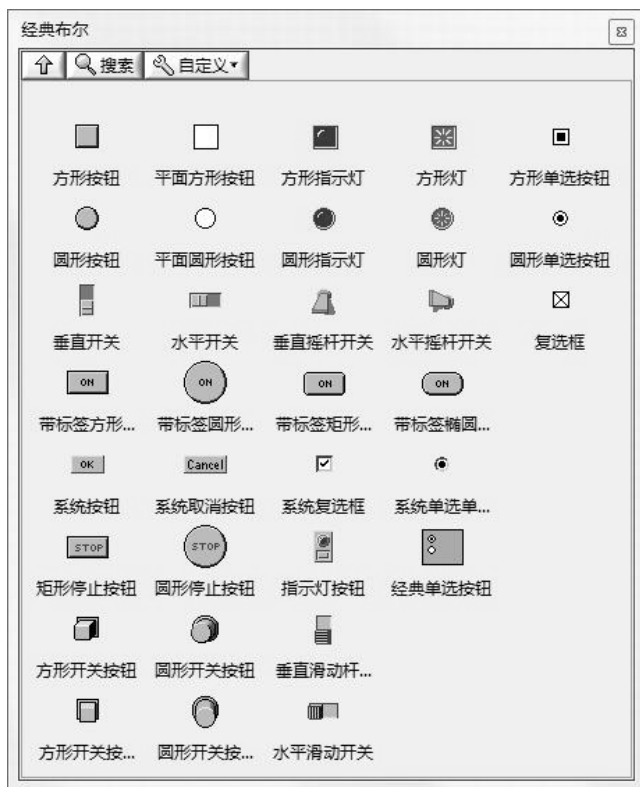


图 3.22 “经典布尔”控件

## 3) 经典字符串及路径控件

经典字符串及路径选板上的控件如图 3.23 所示。



图 3.23 “经典字符串及路径”控件

## 4) 经典数组、矩阵与簇控件

经典数组、矩阵与簇选板上的控件如图 3.24 所示。

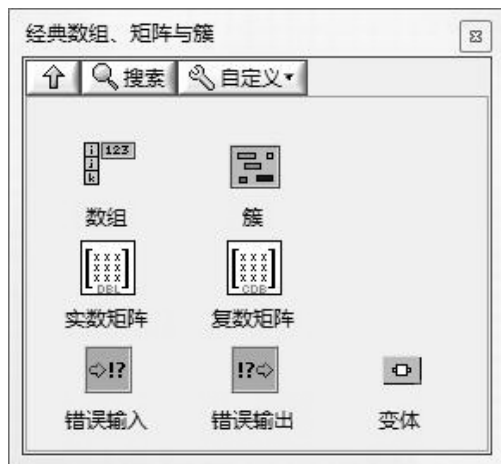


图 3.24 “经典数组、矩阵与簇”控件

### 5) 经典列表、表格和树控件

经典列表、表格和树控件如图 3.25 所示。



图 3.25 “经典列表、表格和树”控件

### 6) 经典图形控件

“经典图形”控件如图 3.26 所示。



图 3.26 “经典图形”控件

## 7) 经典下拉列表及枚举控件

“经典下拉列表及枚举”控件如图 3.27 所示。

## 8) 经典容器控件

“经典容器”控件如图 3.28 所示。



图 3.27 “经典下拉列表及枚举”控件



图 3.28 “经典容器”控件

## 9) 经典 I/O 控件

“经典 I/O”控件如图 3.29 所示。

## 10) 经典引用句柄控件

“经典引用句柄”控件如图 3.30 所示。



图 3.29 “经典 I/O”控件



图 3.30 “经典引用句柄”控件

### 3.2.3 银色控件

银色子选板是 LabVIEW 2013 版及以后的版本新增的新型控件，对比之前的旧版本，银色控件在外观上更形象、逼真，控件类型上与之前版本大致相同，稍有改动。下面详细介绍

该子选板中的控件，如图 3.31 所示。

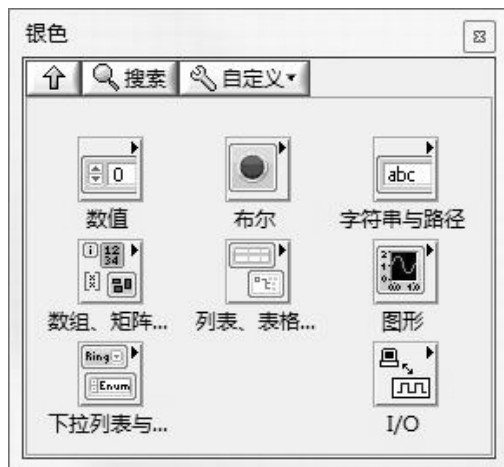


图 3.31 “银色”子选板

### 1) 数值控件

“数值”控件如图 3.32 所示。

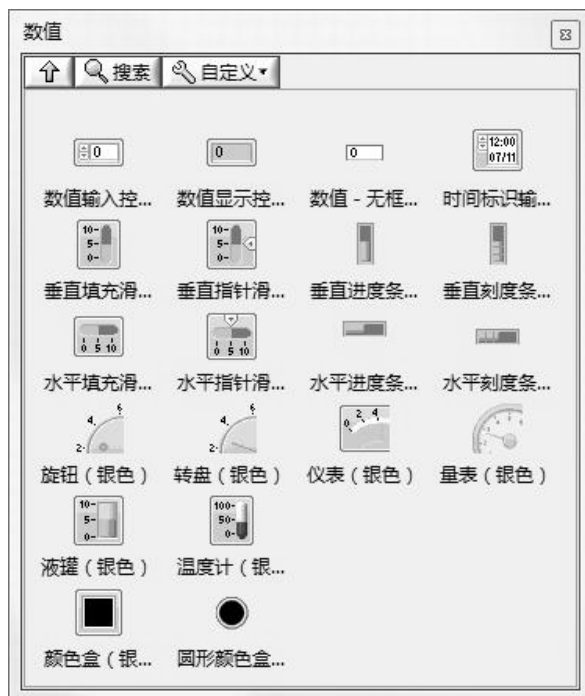


图 3.32 “数值”控件

### 2) 布尔控件

“布尔”控件如图 3.33 所示。

## 3) 字符串与路径控件

“字符串与路径”控件如图 3.34 所示。



图 3.33 “布尔”控件



图 3.34 “字符串与路径”控件

## 4) 数组、矩阵与簇控件

“数组、矩阵与簇”控件如图 3.35 所示。

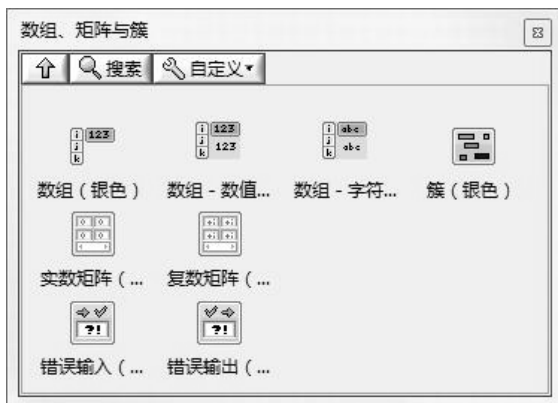


图 3.35 “数组、矩阵与簇”控件

## 5) 列表、表格和树控件

“列表、表格和树”控件如图 3.36 所示。



图 3.36 “列表、表格和树”控件

## 6) 图形控件

“图形”控件如图 3.37 所示。

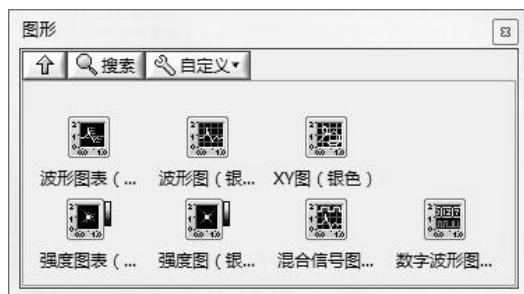


图 3.37 “图形”控件

## 7) 下拉列表与枚举控件

“下拉列表与枚举”控件如图 3.38 所示。



图 3.38 “下拉列表与枚举”控件

## 8) I/O 控件

I/O 控件如图 3.39 所示。



图 3.39 “I/O”控件



### 3.2.4 系统控件

位于系统选板上的系统控件可用在用户创建的对话框中。系统控件专为在对话框中使用而特别设计，包括下拉列表和旋转控件，数值滑动杆、进度条、滚动条、列表框、表格、字符串和路径控件、选项卡控件、树形控件、按钮、复选框、单选按钮等。这些控件仅在外观上与前面板控件不同，颜色与系统设置的颜色一致，如图 3.40 所示。

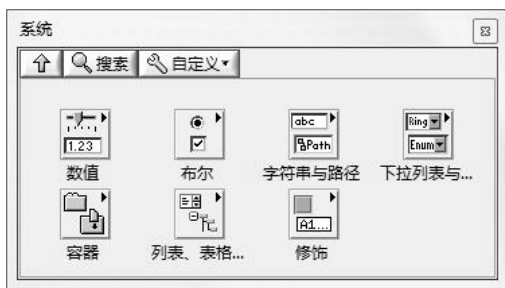


图 3.40 “系统”控件

#### 1) 数值控件

“数值”控件如图 3.41 所示。

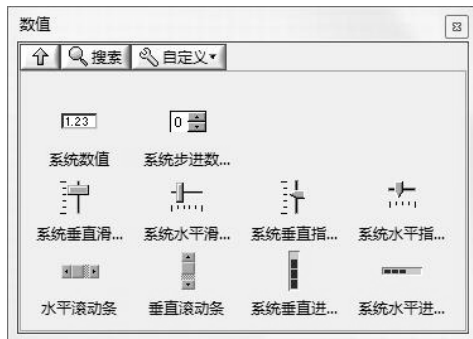


图 3.41 “数值”控件

#### 2) 布尔控件

“布尔”控件如图 3.42 所示。



图 3.42 “布尔”控件

### 3) 字符串与路径控件

“字符串与路径”控件如图 3.43 所示。

### 4) 下拉列表与枚举控件

“下拉列表与枚举”控件如图 3.44 所示。



图 3.43 “字符串与路径”控件



图 3.44 “下拉列表与枚举”控件

### 5) 容器控件

“容器”控件如图 3.45 所示。



图 3.45 “容器”控件

### 6) 列表、表格和树控件

“列表、表格和树”控件如图 3.46 所示。



图 3.46 “列表、表格和树”控件

### 7) 修饰控件

“修饰”控件如图 3.47 所示。



图 3.47 “修饰”控件

### 3.2.5 Express 控件

Express 控件按照输入控件与输出控件的区别进行分类，多为常用控件，如图 3.48 所示。



图 3.48 “Express”控件

#### 1) 数值输入控件

“数值输入控件”如图 3.49 所示。



图 3.49 数值输入控件

#### 2) 按钮与开关控件

“按钮与开关”控件如图 3.50 所示。



图 3.50 “按钮与开关”控件

### 3) 文本输入控件

文本输入控件，如图 3.51 所示。

### 4) Express 用户控件

Express 用户控件如图 3.52 所示。



图 3.51 文本输入控件



图 3.52 “Express 用户控件”控件

### 5) 数值显示控件

数值显示控件如图 3.53 所示。

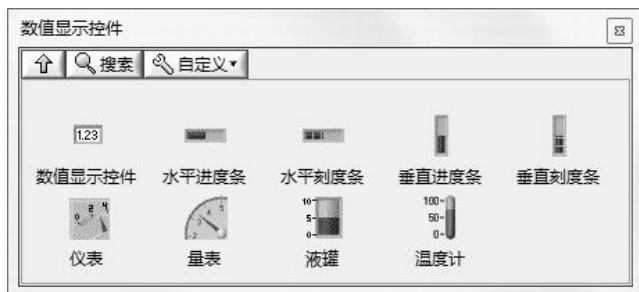


图 3.53 数值显示控件

### 6) 指示灯控件

“指示灯”控件如图 3.54 所示。

## 7) 文本显示控件

“文本显示控件”如图 3.55 所示。



图 3.54 “指示灯”控件



图 3.55 文本显示控件

## 8) 图形显示控件

图形显示控件如图 3.56 所示。



图 3.56 图形显示控件

## 3.2.6 NET 与 ActiveX 控件

位于“.NET”与“ActiveX”选板上的“.NET”与“ActiveX”控件用于对常用的“.NET”或“ActiveX”控件进行操作。可添加更多“.NET”或“ActiveX”控件至该选板，供日后使用，如图 3.57 所示。



图 3.57 “.NET 与 ActiveX”控件

选择“工具→导入→.NET 控件至选板”，弹出“添加.NET 控件至选板”对话框，如图 3.58 所示。

选择“工具→导入→ActiveX 控件至选板”，弹出“添加 ActiveX 控件至选板”对话框，如图 3.59 所示，可分别转换.NET 或 ActiveX 控件集，自定义控件并将这些控件添加至.NET 与 ActiveX 选板。



图 3.58 “添加.NET 控件至选板”对话框



图 3.59 “添加 ActiveX 控件至选板”对话框

创建.NET 对象并与之通信需要安装.NET Framework 1.1 Service Pack 1 或更高版本，建议只在 LabVIEW 项目中使用.NET 对象。如装有 Microsoft .NET Framework 2.0 或更高版本，可使用应用程序生成器生成.NET 互操作程序集。

### 3.3 函数选板、工具选板及 选板可见性设置

#### 1. 函数选板

函数选板只能在编辑程序框图时使用，与控件选板的工作方式大体相同。创建框图程序常用的 VI 和函数对象都包含在该选板中，如图 3.60 所示。

选择“查看 (View)”→“函数选板 (Functions Palette)”选项或右击框图面板空白处就可以显示函数选板。

#### 2. 工具选板

如果启动 LabVIEW 后工具选板没有显示，可以选择“查看 (View)”→“工具选板 (Tools Palette)”选项来显示工具选板，“工具”选板如图 3.61 所示。



图 3.60 函数选板



图 3.61 “工具”选板

在前面板和程序框图中都可以使用工具选板，使用其中不同的工具可以操作、编辑或修饰前面板和程序框图中选定的对象，也可以用来调试程序等。当从工具选板中选择一种工具后，鼠标指针就会变成与该工具相应的形状。如果使用选板最顶端的自动选择工具（Automatic Tool Selection）（或按下 Shift+Tab 快捷键），当光标在前面板或程序框图中移动到相应位置时，LabVIEW 会自动从选板中选择相应的工具。例如，移动到连线端子上时，光标会自动变为线轴形状，表示此时可以连线。

工具选板中各种不同工具的图标及其相应的功能如表 3.2 所示。

表 3.2 工具选板中的各种图标及其相应的功能

名 称	功 能
自动选择工具	如已经打开自动工具选择，光标移到前面板或程序框图的对象上时，LabVIEW 将从工具选板中自动选择相应工具。也可禁用自动工具选择，手动选择工具
操作值工具	改变控件值
定位、调整大小、选择工具	定位、选择或改变对象大小
编辑文本工具	用于输入标签文本或者创建标签

(续表)

名 称	功 能
连线工具	用于在后面板中连接两个对象的数据端口，当用连线工具接近对象时，会显示出其数据端口以供连线之用。如果打开了帮助窗口，那么当用连线工具于某连线上时，会在帮助窗口显示其数据类型
对象快捷菜单工具	当用该工具单击某对象时，会弹出该对象的快捷菜单
滚动窗口工具	使用该工具，无须滚动条就可以自由滚动整个图形
设置/清除断点工具	在调试程序过程中设置断点
探针数据工具	在代码中加入探针，用于调试程序过程中监视数据的变化
获取颜色工具	从当前窗口中提取颜色
设置颜色工具	用来设置窗口中对象的前景色和背景色

### 3. 选板可见性设置

使用控件和函数选板工具栏上的下列按钮，可以查看、配置选板以及搜索控件、VI 和函数，如图 3.62 所示。



图 3.62 选板上的按钮

选板上的各按钮及其功能如表 3.3 所示。

表 3.3 选板上的各按钮及其功能




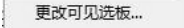
名 称	功 能
返回所属选板 	转到选板的上级目录。单击该按钮并保持光标位置不动，将显示一个快捷菜单，列出当前子选板路径中包含的各个子选板。单击快捷菜单上的子选板名称进入子选板。只有当选板模式设为图标、图标和文本时，才会显示该按钮
搜索 	用于将选板转换至搜索模式，通过文本搜索来查找选板上的控件、VI 或函数。选板处于搜索模式时，可单击返回按钮，退出搜索模式并显示选板
自定义 	用于选择当前选板的视图模式，显示或隐藏所有选板目录，在文本和树形模式下按照字母顺序对各项排序。在快捷菜单中选择选项，可打开选项对话框中的控件/函数选板页，为所有选板选择显示模式。只有当单击选板左上方的图钉标识将选板锁定时，才会显示该按钮
更改可见选板 	调整选板大小。单击此按钮，系统弹出“更改可见选板”对话框，如图 3.63 所示，在该选板中可以更改选板类别、可见性





图 3.63 更改可见选板

## 3.4 项目浏览器

项目浏览器窗口用于创建和编辑 LabVIEW 项目。

选择菜单栏中的“文件→创建项目”命令，打开“创建项目”对话框，如图 3.64 所示，选择“项目”模板，单击“完成”按钮，即可打开“项目浏览器”窗口。

也可以选择“文件→新建”，打开“新建”对话框，双击项目选项，如图 3.65 所示，打开“项目浏览器”窗口。

在默认情况下，项目浏览器窗口包括以下各项。

我的电脑：表示可作为项目终端使用的本地计算机。

依赖关系：用于查看某个终端下 VI 所需要的项。

程序生成规范：包括对源代码发布编译配置以及 LabVIEW 工具包和模板所支持的其他编译形式的配置。如已安装 LabVIEW 专业版开发系统或应用程序生成器，可以使用程序生成规范配置独立应用程序（EXE）、动态链接库（DLL）、安装程序及 Zip 文件。

在项目中添加其他终端时，LabVIEW 会在项目浏览器窗口中创建代表该终端的选项。各个终端也包括依赖关系和程序生成规范，在每个终端下可以添加文件。

可以将 VI 从项目浏览器窗口中拖放到另一个已打开 VI 的程序框图中。在项目浏览器窗口中选择需要作为子 VI 使用的 VI，并把它拖放到其他 VI 的程序框图中。

使用项目属性和方法，可通过编程配置和修改项目以及项目浏览器窗口，如图 3.66 所示。



图 3.64 “创建项目”对话框



图 3.65 “新建”对话框“项目”选项

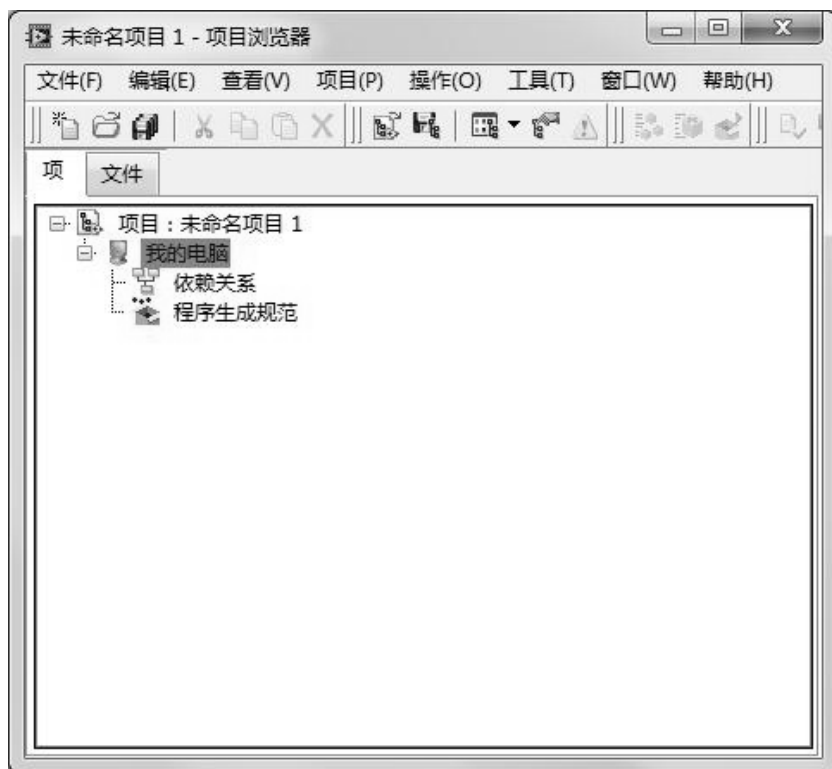


图 3.66 项目管理器窗口

# 第4章 程序框图设计基础

## 4.1 程序框图结构

创建前面板窗口后，需要为代码添加图形化函数，用于控制前面板对象。程序框图窗口中是图形化源代码。

程序框图由连线端、子 VI、函数、常量、结构和连线构成，连线可在其他的程序框图对象间传递数据。

### 1) 接线端

前面板上的对象在程序框图中显示为接线端。接线端是在前面板和程序框图之间交换信息的输入、输出端口。接线端类似于文本编辑语言中的参数和常数。

接线端的类型包括输入/显示控件接线端和节点接线端。输入控件接线端和显示控件接线端属于前面板输入控件和显示控件。在前面板控件中输入的数据将通过控件接线端传输至程序框图中，然后再进行数据的加减运算。加减运算结束后，将输出新的数据值。数据将传输至显示控件接线端，更新前面板显示控件中的数据。

### 2) 子 VI

子 VI 是建立在其他 VI 内部的 VI 或者函数选板上的 VI。子 VI 类似于文本编程语言中的函数。

任何 VI 都可以用做子 VI。双击程序框图中的子 VI，将出现该子 VI 的前面板窗口。前面板包括输入控件和显示控件。程序框图既包括连线、图标、函数，也可能有子 VI 和其他 LabVIEW 对象。

### 3) 图标

每个 VI 前面板和程序框图窗口的右上角都有一个图标。图标是 VI 的图形化表示，图标可以包括图像。如果将一个 VI 当作子 VI 使用，程序框图上将显示代表该子 VI 的图标。默认图标中有一个数字，表明 LabVIEW 启动后打开新 VI 的个数。

要将一个 VI 当作子 VI 使用，必须创建连线板，连接板是一组与 VI 中的输入控件和显示控件对应的接线端，类似于文本编程语言中的函数调用参数列表。右键单击前面板窗口右上角的图标即可访问连线板。在程序框图窗口中无法通过图标访问连线板。

子 VI 也有可能是 Express VI。Express VI 所需要的连线节点最小，可以用对话框对它们进行设置。使用 Express VI 可以实现一些常规的测量任务，也可以将设置好的 Express VI 保

存为一个子 VI。

LabVIEW 使用彩色图标以区分 Express VI 和程序框图上的其他 VI。程序框图中 Express VI 的图标为浅蓝色背景，而子 VI 为黄色背景。

#### 4) 函数

函数是 LabVIEW 中最基本的操作元素，它们没有前面板或程序框图窗口，但有连线板。双击一个函数只是选择该函数，函数图标的背景为浅黄色。

程序框图中所需要的函数与 VI 均放置在函数选板中，根据不同需求将函数进行分类，放置在不同的选项下，如图 4.1 所示。



图 4.1 函数分类


#### 5) 常量

常量是只在程序框图中显示且有别于控件的数值，作用与控件类似，但不在前面板中显示。

#### 6) 结构

结构是过程控制元素，例如条件结构、For 循环或 While 循环。

## 7) 连线

当光标变成“”图标时，表明正在使用连线工具。连线工具用于连接程序框图上的对象。

通过连线可以在程序框图对象之间传递数据。每根连线都只有一个数据源，但可以与多个读取该数据的 VI 和函数连接。不同数据类型的连线有不同的颜色、粗细和式样。

## 4.2 数学函数与 VI

在函数选板中选择“数学”命令，打开图 4.2 所示的“数学”子选板，在该面板中包括基本的几何运算函数、数组几何运算函数，不同类型的数值常量等，另外，还包括 6 个带子选板的选项。



图 4.2 “数学”子选板

### 4.2.1 数值函数

选择“数学→数值”命令，打开图 4.3 所示的“数值”子选板，在该面板中包括基本的几何运算函数、数组几何运算函数，不同类型的数值常量等，另外，还包括 6 个带子选板的选项。

#### 1) 转换

选择“数学”→“数值”→“转换”选项，打开图 4.4 所示的子选板。该面板中函数的功能主要是转换数据类型。在 LabVIEW 中，一个数据从产生开始决定它的数据类型，不同类型的数据无法进行运算操作，因此当两个不同类型的数据进行运算时，需要转换，只有相同类型的数据才能进行运算，否则数据连线上将显示错误的信息。

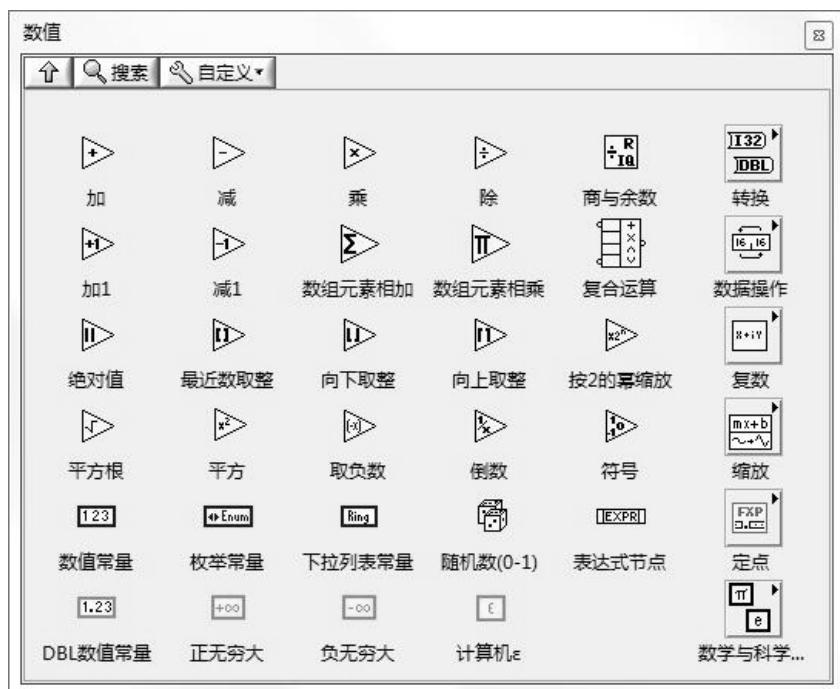


图 4.3 “数值”子选板



图 4.4 “转换”子选板

## 2) 数据类型

选择“数据操作”选项，打开图 4.5 所示的子选板，面板中的函数用于改变 LabVIEW 使用的数据类型。



图 4.5 “数据操作”子选板

## 3) 复数

选择“复数”选项，打开图 4.6 所示的子选板。该面板中的函数用于根据两个指数坐标或极坐标的值创建复数或将复数分为直角坐标或极坐标的两个分量，具体介绍如下所述。



图 4.6 “复数”子选板

- 复共轭：计算  $x+iy$  的复共轭。
- 复数和极坐标转换：使复数分解为极坐标分量。
- 复数至实部虚部转换：使复数分解为直角分量。
- 极坐标至复数转换：通过极坐标分量的两个值创建复数。
- 极坐标至实部虚部转换：使复数从极坐标系转换为直角坐标系。
- 实部虚部至复数转换：通过直角分量的两个值创建复数。
- 实部虚部至极坐标转换：使复数从直角坐标系转换为极坐标系。

## 4) 缩放

选择“缩放”选项，打开图 4.7 所示的子选板。该面板中的 VI 可将电压读数转换为温度或其他应变单位。





图 4.7 “缩放”子选板

### 5) 定点

选择“定点”选项，打开图 4.8 所示的子选板。该面板中的函数可对定点数字的溢出状态进行操作。

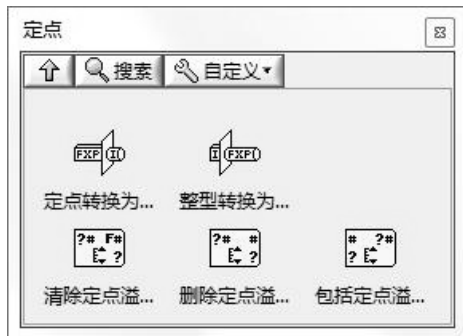


图 4.8 “定点”子选板

### 6) 数学与科学常量

选择“数学与科学常量”选项，打开图 4.9 所示的子选板。该面板中函数的功能主要为特定常量，下面介绍常量代表的数值。

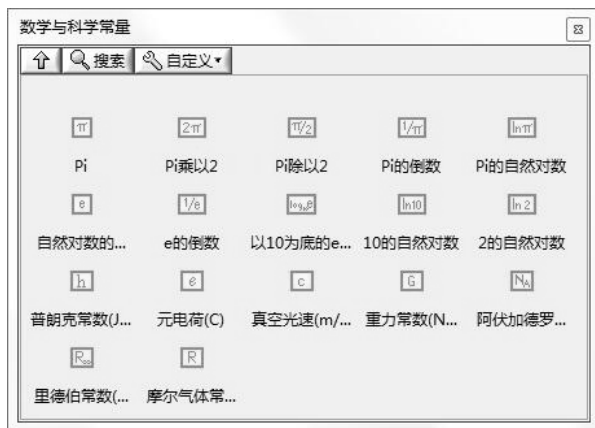


图 4.9 “数学与科学常量”子选板

- 阿伏伽德罗常数 ( $1/\text{mol}$ ):  $6.022\ 141\ 79\text{e}23$ 。
- 以 10 为底的 e 的对数:  $0.434\ 294\ 481\ 903\ 251\ 83$ 。
- 元电荷 (C):  $1.602\ 176\ 487\text{e}-19$ 。
- 重力常数 ( $\text{N}\cdot\text{m}^2/\text{kg}^2$ ):  $6.674\ 28\text{e}-11$ 。
- 摩尔气体常数 ( $\text{J}/(\text{mol}\cdot\text{K})$ ):  $8.314472$ 。
- 自然对数的底数:  $2.718\ 281\ 828\ 459\ 045\ 2$ 。
- Pi 的自然对数:  $1.144\ 729\ 885\ 849\ 400\ 2$ 。
- 2 的自然对数:  $0.693\ 147\ 180\ 559\ 945\ 31$ 。
- 10 的自然对数:  $2.302\ 585\ 092\ 994\ 059\ 7$ 。
- Pi:  $3.141\ 592\ 653\ 589\ 793\ 2$ 。
- Pi 除以 2:  $1.570\ 796\ 326\ 794\ 896\ 6$ 。
- Pi 乘以 2:  $6.283\ 185\ 307\ 179\ 586\ 5$ 。
- 普朗克常数 ( $\text{J}\cdot\text{s}$ ):  $6.626\ 068\ 96\text{e}-34$ 。
- e 的倒数:  $0.367\ 879\ 441\ 171\ 442\ 32$ 。
- Pi 的倒数:  $0.318\ 309\ 886\ 183\ 790\ 67$ 。
- 里德伯常数 ( $1/\text{m}$ ):  $10\ 973\ 731.568\ 527$ 。
- 真空光速 ( $\text{m/s}$ ):  $299\ 792\ 458$ 。

## 4.2.2 初等与特殊函数

选择“数学”→“初等与特殊函数”，打开图 4.10 所示的“初等与特殊函数”子选板，该面板用于常见数学函数的运算。

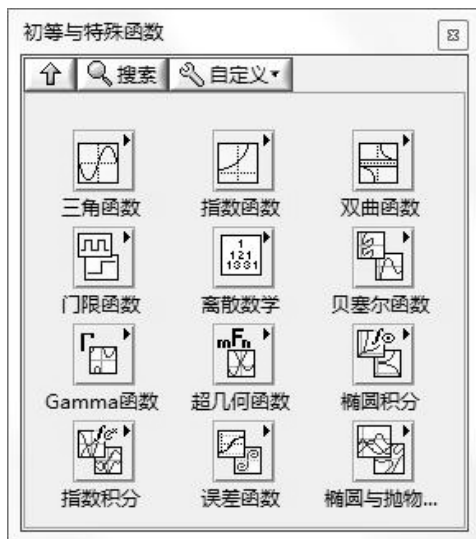
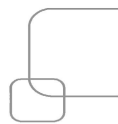


图 4.10 “初等与特殊函数”子选板

下面介绍各函数的功能。



- Gamma 函数 VI：该类函数专用于计算 Gamma 相关函数。
- 贝塞尔函数 VI：该类函数专用于计算贝塞尔函数。
- 超几何函数 VI：该类函数专用于计算基于微分方程的超几何函数。
- 离散数学函数 VI：此类基本函数用于计算如组合数学及数论领域的离散数学函数。
- 门限函数 VI：此类初等函数用于计算一些常用的周期波在给定点上的采样值。
- 三角函数：该类初等函数用于计算三角函数及其反函数。
- 双曲函数：此类基本函数用于计算双曲函数及其反函数。
- 椭圆积分 VI：该类函数专用于计算完全或不完全椭圆积分。
- 椭圆与抛物函数 VI：该类函数专用于计算特定的椭圆积分或韦伯函数。
- 误差函数 VI：该类函数专用于计算误差相关函数。
- 指数函数：该类初等函数用于计算指数函数和对数函数。
- 指数积分 VI：该类函数专用于计算指数积分。

### 4.2.3 函数快捷命令

一般的函数或 VI 包括图标、输入端和输出端。图标以简单的图画来显示其作用；输出、输入端用来连接控件、常量或其余函数，也可空置。在函数右键快捷菜单中显示函数的操作，如图 4.11 所示。



图 4.11 快捷菜单

在不同的函数或 VI 上显示的快捷菜单也不同。下面简单介绍快捷菜单中常用的命令。

- (1) 显示项：该项包括函数的基本参数：标签与连接线，图标一般以图例的形式显示，接线端以直观的方式显示输入、输出端的个数。
- (2) 断点：利用该命令，可启用、禁用断点。
- (3) 数值选板：在该子选板中选择函数与 VI。
- (4) 创建：选择该命令，弹出快捷菜单，可在函数输入、输出端创建“常量”、“输入控

件”、“输出控件”、“局部变量”、“引用”、“属性节点”、“调用节点”等对象。

(5) 替换：将该函数或 VI 替换为其余函数，此操作适用于绘制完成的 VI，各函数已互相连接，若在该处删除原函数、添加新函数，容易导致连线发生错误，因此在此种情况下使用“替换”命令，一般要替换的函数与原函数输入端、输出端个数相同，不易发生连线错误的现象。

(6) 属性：选择该命令，弹出“对象属性”设置对话框，如图 4.12 所示。该对话框与前面板中控件的属性设置对话框相似，这里不再赘述。

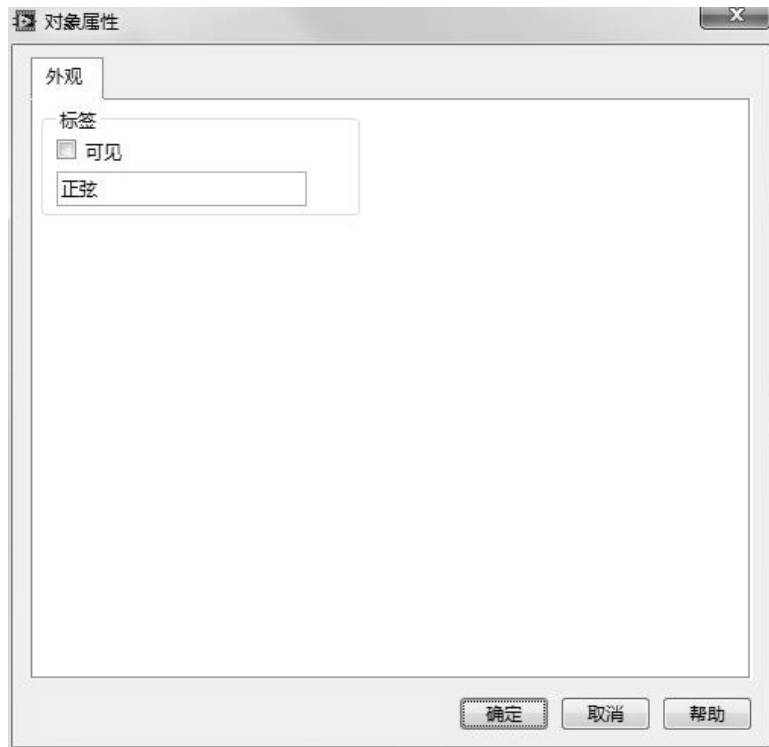


图 4.12 属性设置对话框

## 4.3 VI 的设计

### 4.3.1 创建 VI 前面板

在 VI 前面板窗口的空白处单击鼠标右键，或者选择菜单栏中的“查看”→“控件选板”，弹出控件选板。

在控件选板中，选择“银色”→“数值”→“数值输入控件”，并将其放在前面板窗口的适当位置上。单击数值输入控件的标签，把名称修改为“A”，如图 4.13 所示。

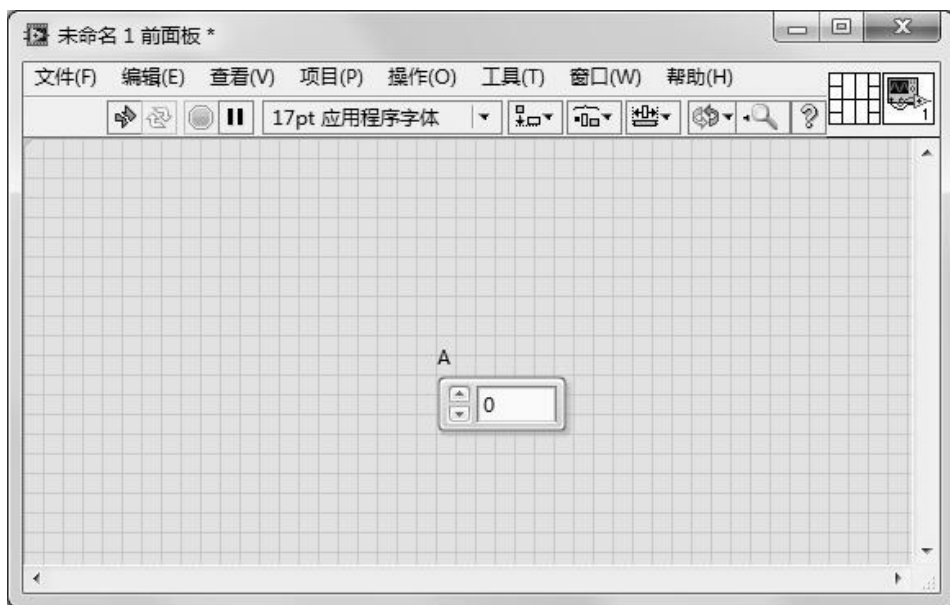


图 4.13 创建数值输入控件 A

此时，在框图中就会出现一个名称为“A”的端口图标与输入量 A 相对应，如图 4.14 所示。

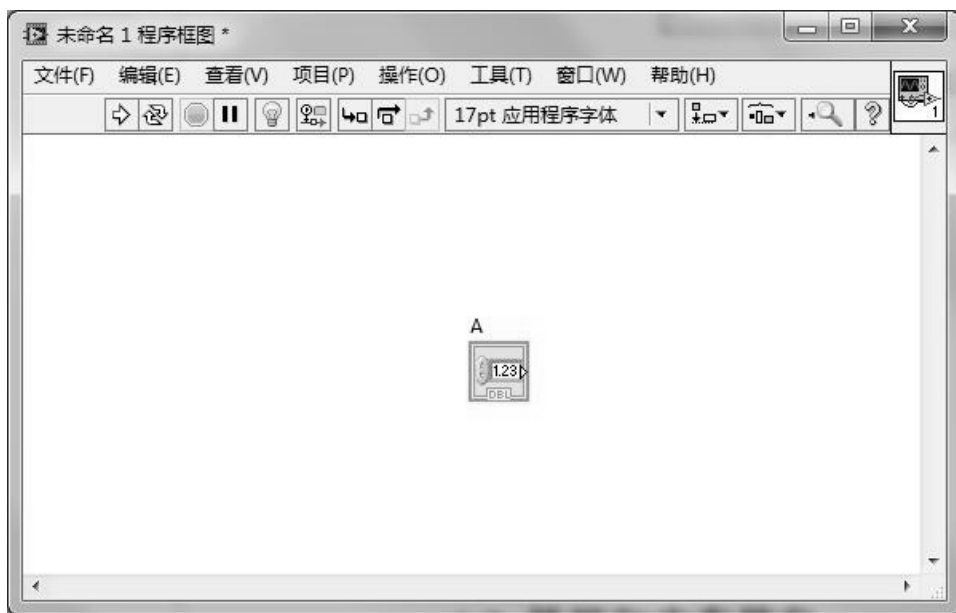


图 4.14 数值输入控件 A 的端口图标

在控件选板中，选择“银色”→“数值”→“水平指针滑动杆”，并将其放在前面板窗口的适当位置上。单击数值输入控件的标签，把名称修改为“B”，如图 4.15 所示，创建数值输入控件 B。

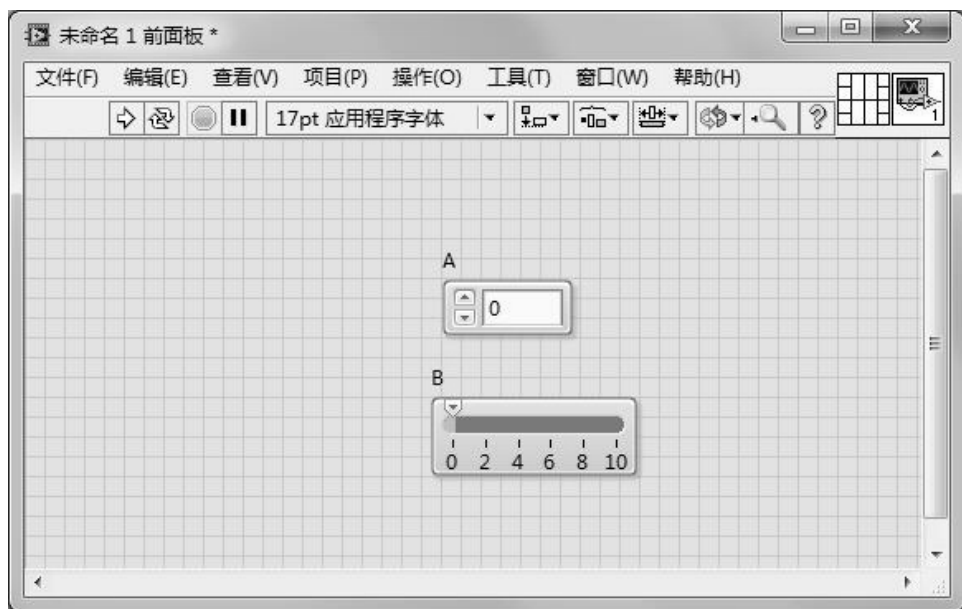


图 4.15 创建数值输入控件 B

同时，在框图中就会出现一个名称为 B 的端口图标与输入量 B 相对应，如图 4.16 所示。

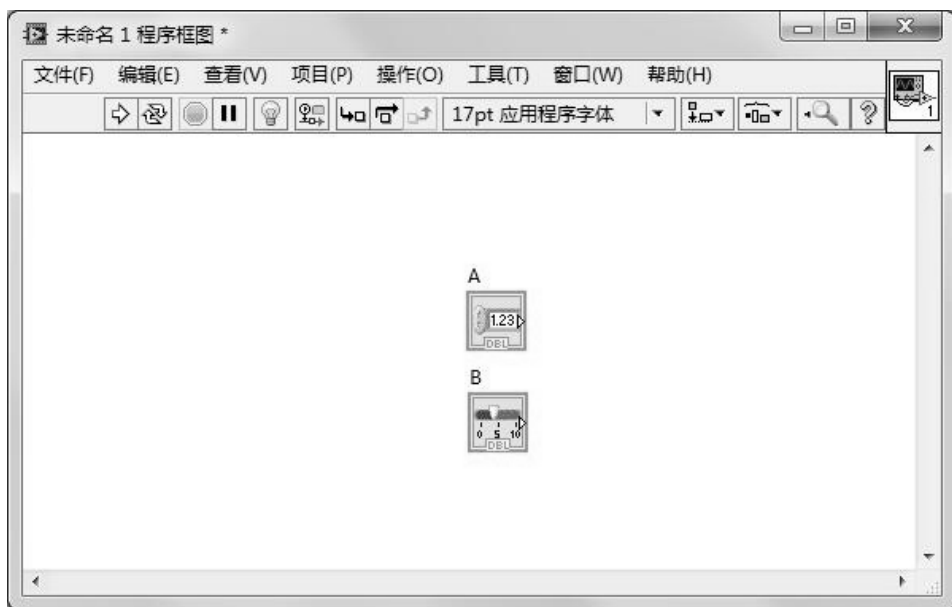


图 4.16 输入控件 B 的端口图标

在控件选板中，选择“银色”→“数值”→“仪表（银色）”，将其放置在前面板窗口的适当位置上，用文本编辑工具单击数值输出控件的标签，把名称修改为 C。

此时，就完成了 VI 前面板的创建，如图 4.17 所示。

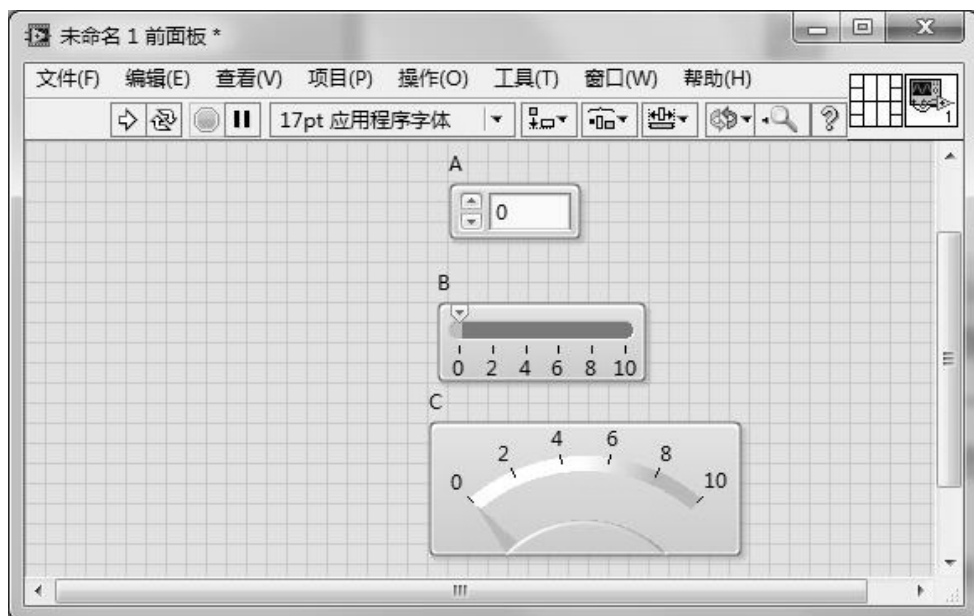


图 4.17 VI 的前面板

### 4.3.2 创建程序框图

在前面板窗口中，选择菜单栏中的“窗口”→“显示程序框图”命令，将前面板窗口切换到程序框图窗口，此时在程序框图中会看到 3 个名称分别为 A、B 和 C 的端口图标，如图 4.18 所示。这 3 个端口图标与前面板的 3 个对象一一对应。

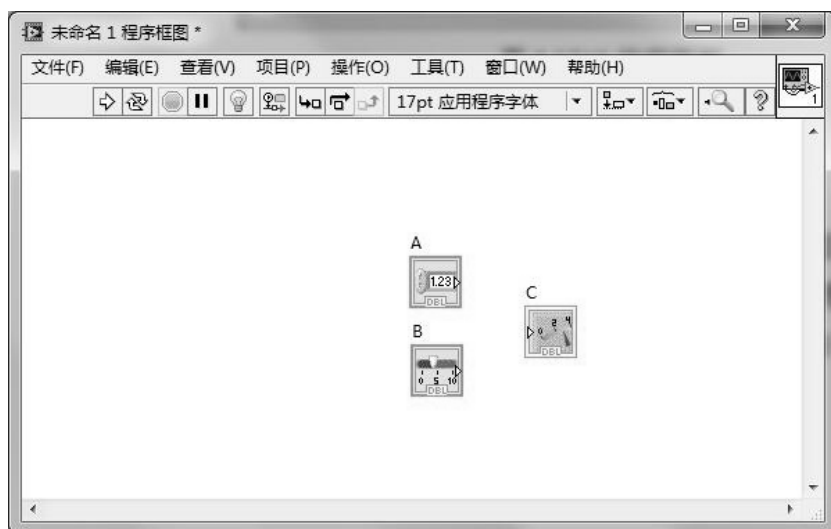


图 4.18 前面板对象的端口图标

在程序框图窗口中的空白处单击鼠标右键，或在框图程序窗口的菜单栏中选择“查看”→“函数选板”命令，弹出函数选板。

在函数选板中选择“编程”→“数值”→“加”节点。用鼠标将“加”节点的图标拖到程序窗口的适当位置。这样，就完成了“加”节点的创建工作，如图 4.19 所示。



图 4.19 创建“加”节点

### 4.3.3 对象连接

完成了框图程序所需要的端口和节点的创建之后，下面的工作就是用数据连线将这些端口和图标连接起来，形成一个完整的框图程序。

用连线工具将端口 A 和 B 分别连接到“加”节点的两个输入端口 x 和 y 上，将端口 C 连接到“乘”节点的输出端口 xy 上。完成了数据连线的创建之后，将鼠标切换到对象操作工具状态，适当调整各图标及数据连线的位置，使之整齐美观，完成的程序框图如图 4.20 所示。

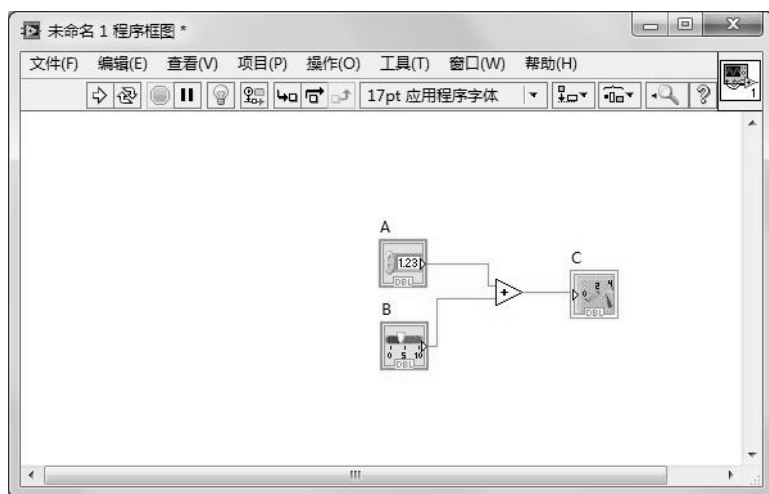


图 4.20 连线结果



程序框图中的函数与 VI 的选取只是设计的第一步，只有进行正确连线，将所有对象按照数据流正确进行连接，才能运行出正确的结果。

关于对象连接有以下几种情况。

### 1) 自动连接对象

自动连线指放开鼠标将对象放置在程序框图上时，LabVIEW 会自动进行连线，也可以对程序框图上已经存在的对象进行自动连线。

LabVIEW 会对最匹配的连线端进行连线，对不匹配的连线端则不予连线。在默认状态下，从“函数”选板中选择一个对象，或者按<Ctrl>键并拖动对象来复制一个已经存在于程序框图上的对象时，自动连线方式被启用。在默认状态下，用定位工具移动程序框图上已经存在的对象时，自动连线被取消。

选择菜单栏中的“工具”→“选项”命令，然后从类别列表中选择程序框图，在“连线”选项下可调整自动连线设置，如图 4.21 所示。

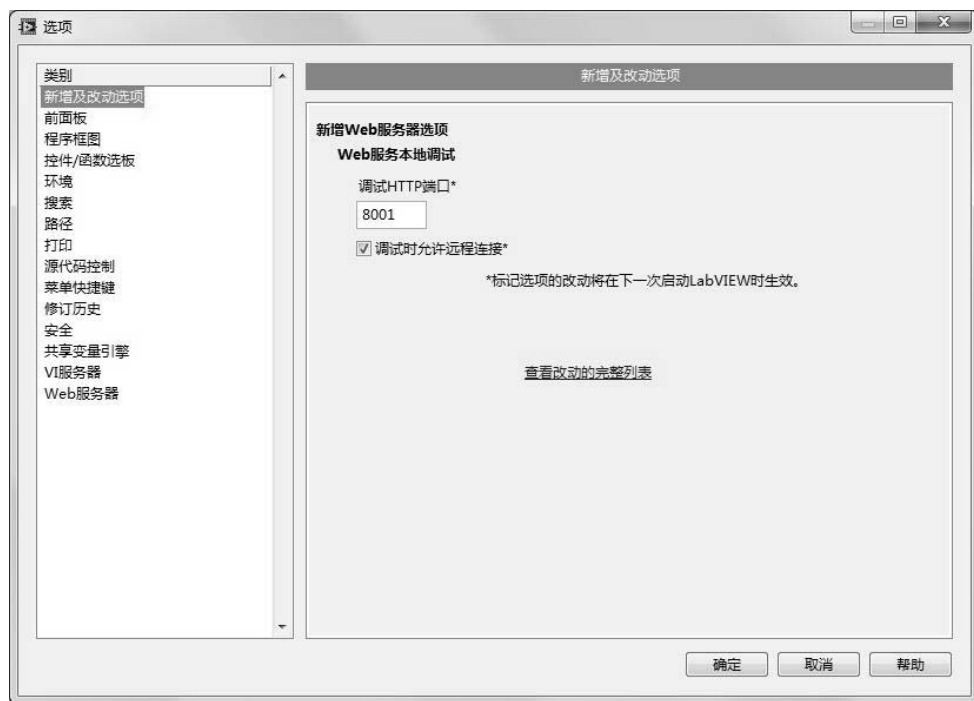


图 4.21 “连线”选项

### 2) 手动连接对象

将连线工具移至接线端时，将出现含有接线端名称的提示框。另外，即时帮助窗口和图标上的接线端都将闪烁，以帮助用户确认正确的接线端。将连线工具移至第一个接线端上并单击，然后将光标移动到第二个接线端再次单击，就可以在这两个对象之间创建连线。

连线结束之后，右键单击连线，从快捷菜单中选取“整理连线”，可使 LabVIEW 自动选择连线路径，按<Ctrl+B>组合键可删除在程序框图中的所有断线。

### 3) 错误连线

连线用于连接多个接线端，从而在 VI 中传递数据。断线显示为一条中间带有红色“×”的黑色虚线，如图 4.22 所示。



图 4.22 断线连接



出现断线有多种原因，图中断线产生的原因为两个对象连接数据类型不兼容。下面介绍断线产生的大致原因。

- (1) 连线连接的输入端和输出端必须与连线上传输的数据类型兼容。
- (2) 连线的方向必须正确。连线必须有一个输入和至少一个输出。例如，不能在显示控件间连线。决定连线兼容性的因素包括输入/显示控件的数据类型和接线端的数据类型。




## 4.3.4 运行 VI

在 LabVIEW 中，用户可以通过四种方式来运行 VI，即运行、连续运行、停止运行和暂停运行，下面介绍这四种运行方式的使用方法。



### 1) 运行 VI

在前面板窗口或程序框图窗口的工具栏中单击“运行”按钮, 即可运行 VI。使用这种方式运行 VI，VI 只运行一次，当 VI 正在运行时，“运行”按钮会变为（正在运行）状态。


### 2) 连续运行 VI

在工具栏中单击“连续运行”按钮, 可以连续运行 VI。连续运行的意思是指一次 VI 运行结束后，连续重新运行 VI。当 VI 正在连续运行时，“连续运行”按钮会变为（正在连续运行）状态。单击按钮可以停止 VI 的连续运行。

### 3) 停止运行 VI

当 VI 处于运行状态时，在工具栏中单击“终止执行”按钮, 可强行终止 VI 的运行。这项功能在程序的调试过程中非常有用，当不小心使程序处于死循环状态时，用该按钮可安全地终止程序的运行。当 VI 处于编辑状态时，“终止执行”按钮处于（不可用）状态，此时的按钮是不可操作的。

### 4) 暂停运行 VI

在工具栏中单击“暂停”按钮, 可暂停 VI 的运行，再次单击该按钮，可恢复 VI 的运行。

### 4.3.5 设置图标

一个完整的 VI 是由前面板、程序框图、图标和连接端口组成的。图标的设计图案不是随手涂鸦，它是以最直观的符号或图形让读者明白图标所代表 VI 的含义。

下面介绍几种常见 VI 的图标。

双击前面板右上角的图标，弹出图 4.23 所示的“图标编辑器”对话框，在该对话框中编辑图标，该对话框中包括菜单栏、选项卡、工具栏及绘图区。

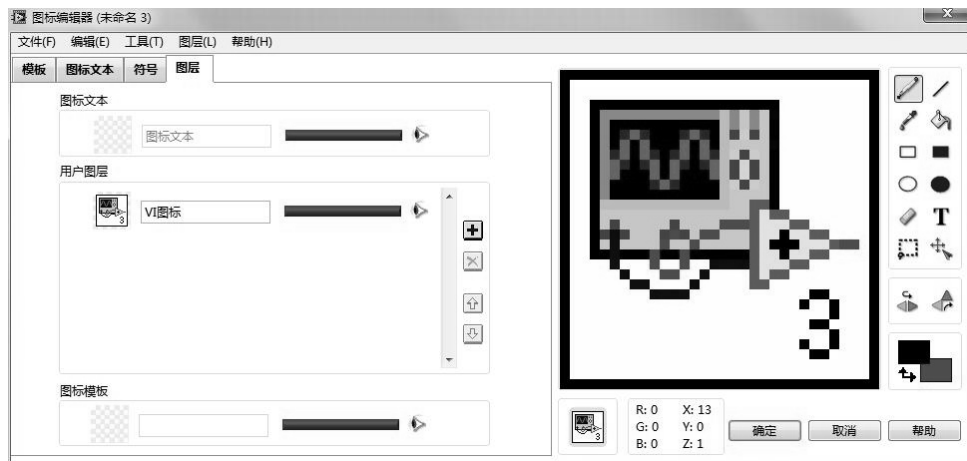


图 4.23 “图标编辑器”对话框

(1) 该对话框包括 4 个选项卡。

- 在“模板”选项卡中选择需要的模板，导入绘图区，方便简洁。
- 在“图标文本”选项卡中设置图标中要输入的文字、符号等，同时可设置输入的文字字体、颜色和样式。
- 在“符号”选项卡中显示多种图形符号，可作为图标编辑的基础部件，按照要求选择基本图形，装饰图标，如图 4.24 所示。

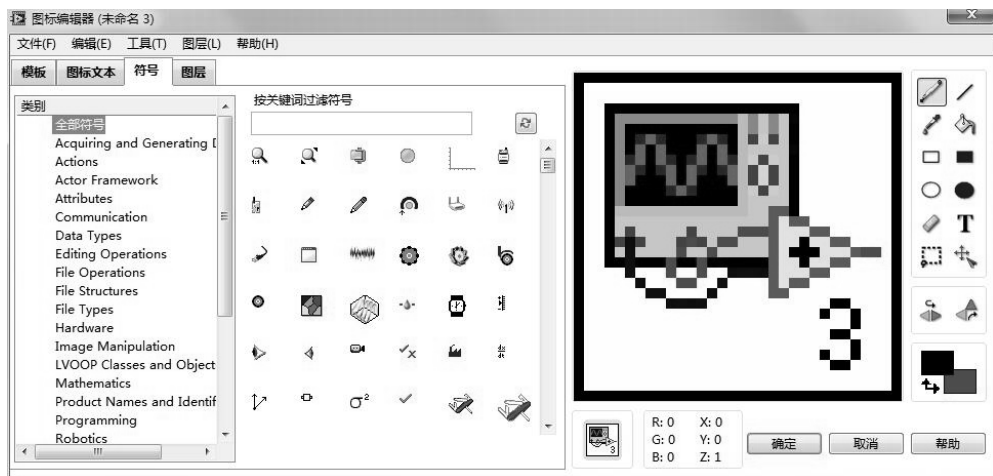


图 4.24 装饰图标

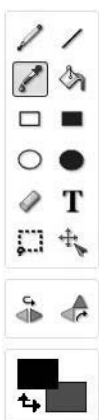



图 4.25 工具栏

- 在“图层”选项卡中设置图标对象的图层，图形或文字的前后次序同样影响图标的显示结果。

(2) 工具栏中包括 3 部分，如图 4.25 所示，从上到下依次为绘图、布局 and 颜色。

- 绘图部分包括 12 种工具，可利用这些工具在绘图区绘制图形。
- 布局部分包括两种工具：水平翻转和垂直翻转，合理使用该工具，使图形达到所需要的效果。
- 颜色部分可设置绘制的图形颜色。

(3) 绘图区中一般显示系统默认的图形，在设置图标过程中，首先应选择  按钮，删除右侧褐色边框内部的图标，如图 4.26 所示。

在空白黑框中进行图标绘制，如果太粗，也可以删除黑色边框。

打开“图标文本”选项卡，在“第一行”文本栏中输入“A+B”，其余参数为默认设置，如图 4.27 所示。

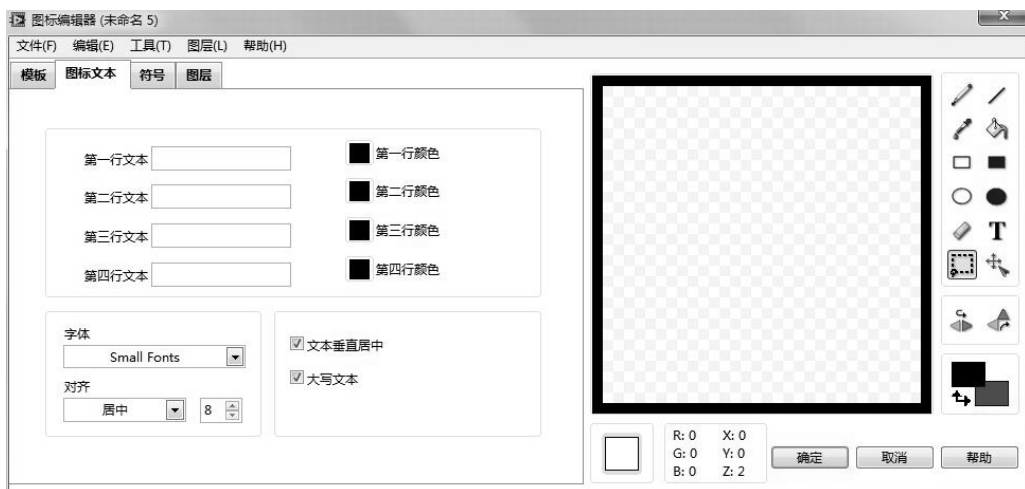


图 4.26 修改图标

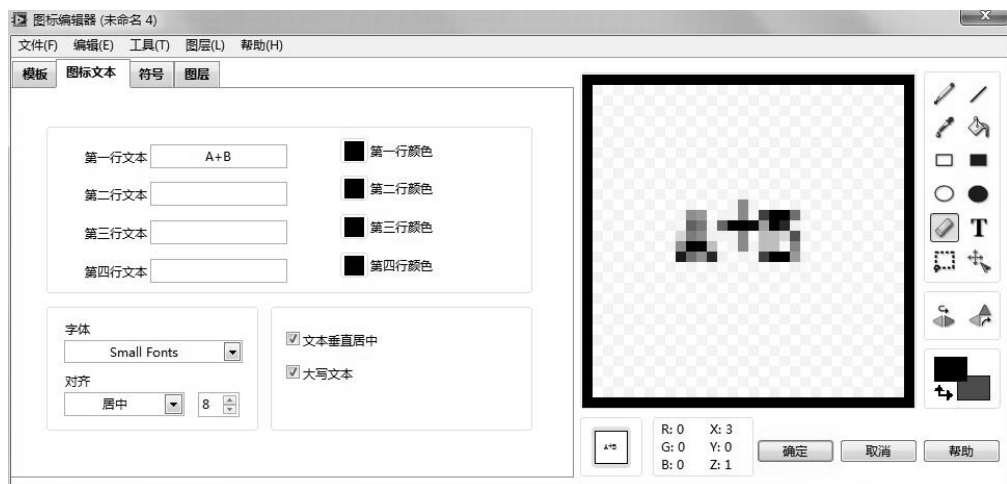
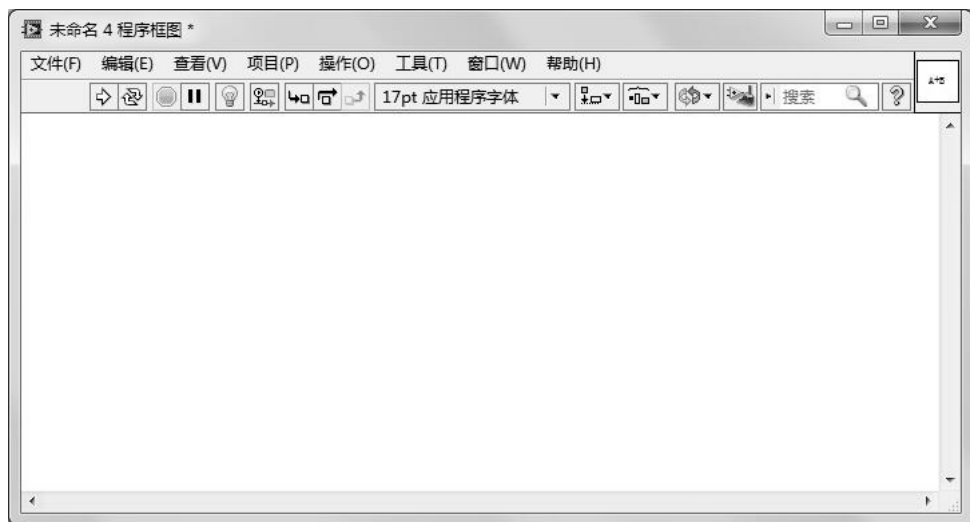


图 4.27 “图标编辑器”窗口

单击“确定”按钮，退出对话框。前面板与程序框图结果如图 4.28 所示，可以看到右上角的图标发生变化。




(a)



(b)

图 4.28 修改图标后的程序框图

至此就完成了创建一个 VI 的创建。在输入量 A 和 B 中分别输入适当的数字值，然后单击前面板窗口工具条中的“运行”按钮，就可以在输出控件 C 中得到计算结果。

## 4.4 调 试 VI

本节将讨论 LabVIEW VI 的基本调试方法，LabVIEW 提供了有效的编程调试环境，同时提供了许多与优秀的交互式调试环境相关的特性。这些调试特性与图形编程方式保持一

致，可通过图形方式访问来调试功能。通过加亮执行、单步、断点和探针帮助用户跟踪经过 VI 的数据流，从而使调试 VI 更容易。实际上用户可以观察 VI 执行时的程序代码。

## 1. 纠正 VI 的错误


由于编程错误而使 VI 不能编译或运行时，工具条上将出现“列出错误 (Broken run)”按钮。典型的编程错误出现在 VI 开发和编程阶段，而且一直保留到将框图中的所有对象都正确地连接起来之前。单击“列出错误”按钮，可以列出所有的程序错误，列出所有程序错误的信息框称为“错误列表”。具有断线的 VI 的“错误列表”对话框如图 4.29 所示。



图 4.29 “错误列表”对话框

当运行 VI 时，警告信息让用户了解潜在的问题，但不会禁止程序运行。如果想知道有哪些警告，在“错误列表”对话框中选择“显示警告”复选框，这样，每当出现警告情况时，工具栏上就会出现警告按钮。

如果程序中有阻止程序正确执行的任何错误，通过在错误列表中选择错误项，然后单击“显示错误”按钮，可搜索特定错误的源代码。这个过程加亮框图上报告错误的对象。另外，在错误列表中单击错误也将加亮报告错误对象。

在编辑期间导致中断 VI 的一些最常见的原因如下：

- (1) 要求输入的函数端子未连接。例如，算数函数的输入端如果未连接，将报告错误。
- (2) 由于数据类型不匹配或存在散落、未连接的线段，使框图包含断线。

## 2. 高亮显示程序执行过程



通过单击“高亮显示执行过程”按钮, 可通过动画演示 VI 框图的执行情况, 该按钮位于图 4.30 所示的程序框图工具栏中。



图 4.30 运行调试工具栏

程序框图的“高亮显示执行过程”按钮对于调试很有帮助。当单击该按钮时, 按钮变为闪亮的灯泡, 指示当前程序执行时的数据流情况。任何时候单击该按钮将返回正常运行模式。

“高亮显示执行过程”普遍用于单步执行模式下跟踪框图中数据流的情况, 目的是了解数据在框图中是如何流动的。应该注意的是, 当使用高亮显示执行过程特性时, VI 的执行时间将大大增加。数据流动画用“气泡”来指示沿着连线运动的数据, 演示从一个节点到另一个节点的数据运动。另外, 在单步模式下, 要执行的下一个节点将一直闪烁, 直到单击单步按钮为止。

## 3. 单步通过 VI 及其子 VI

为了进行调试, 可以一个节点接着一个节点地执行程序框图, 这个过程称为单步执行。要在单步模式下运行 VI, 在工具条上按任何一个单步调试按钮, 然后继续进行下一步即可。单步按钮位于程序框图面板左上角的工具栏上, 如图 4.31 所示。所按的单步按钮决定下一步从哪里开始执行。单击“单步步入”或“单步步过”按钮时执行完当前节点后前进到下一个节点。例如, 如果节点是子 VI, 单击“单步步过”按钮, 则执行子 VI 并前进到下一个节点, 但不能看到子 VI 节点内部是如何执行的。要单步通过子 VI, 应选择“单步步入”按钮。



图 4.31 单步执行按钮

单击“单步步入”按钮完成框图节点的执行。当按任何一个单步按钮时, 也相当于按了“暂停”按钮。在任何时候通过释放“暂停”按钮可返回正常执行的情况。

值得提示的是, 如果将光标放置到任何一个单步按钮上, 将出现一个提示条, 显示如果按该按钮时下一步将要执行的内容描述。

当单步通过 VI 时, 可能需要高亮显示执行过程, 以便数据流过时可以跟踪数据。在单步和高亮显示执行过程模式下执行子 VI 时, 子 VI 的框图窗口显示在主 VI 程序框图的上, 接着可以单步通过子 VI 或让其自动执行。

没有单步或高亮显示执行过程的 VI 可以节省开销。在一般情况下这种编译方法可以减少内存需求并提高性能。其实现方法是, 在菜单栏中选择“文件”→“VI 属性”命令, 弹出“VI 属性”对话框。在“类别”下拉列表框中选择“执行”选项, 取消“允许调试”复选框来隐藏“高亮显示执行过程”及“单步执行”按钮, 如图 4.32 所示。

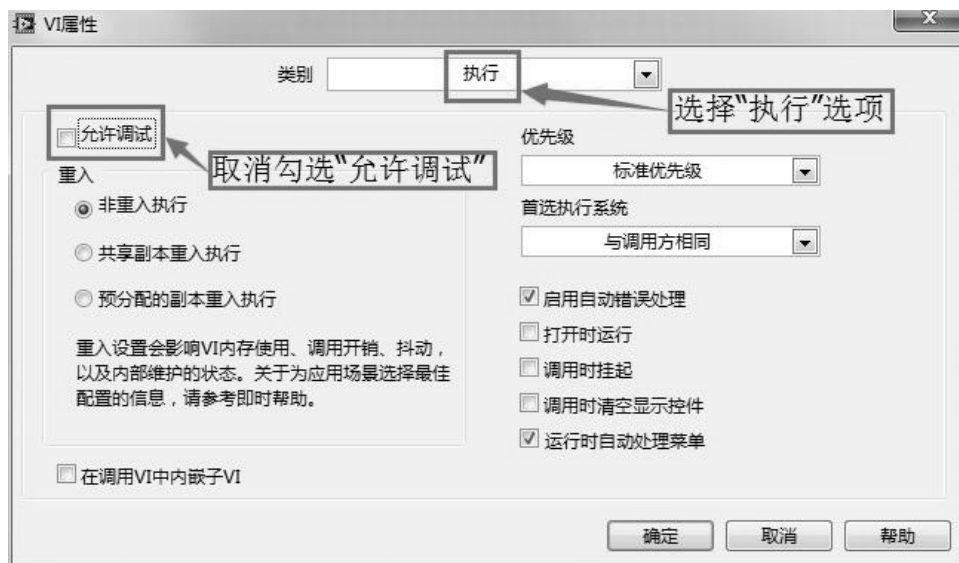


图 4.32 “VI 属性”对话框

## 4.5 子 VI

子 VI 相当于常规编程语言中的子程序，在 LabVIEW 中，用户可以把任何一个 VI 当作子 VI 来调用。因此在使用 LabVIEW 编程时，也应与其他编程语言一样，尽量采用模块化编程的思想，有效地利用子 VI，简化 VI 框图程序的结构，使其更加简单，易于理解，以提高 VI 的运行效率。

子 VI 利用连接端口与调用它的 VI 交换数据。实际上，创建完成一个 VI 后，再按照一定的规则定义好 VI 的连接端口，该 VI 就可以作为一个子 VI 来使用了。

### 1. 创建子 VI



图 4.33 快捷菜单

在完成一个子 VI 的创建后，将其作为子 VI 调用的主要工作就是定义 VI 的连接端口。

在 VI 前面板的右上角显示图标与接线端两个小图形，在程序框图中只显示图标。接线端的位置就会显示一个连接端口。

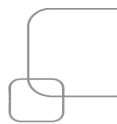
第一次打开连线板时，LabVIEW 会自动根据前面板中的输入和输出控件建立相应个数的端口。当然，这些端口并没有与输入或显示控件建立起关联关系，需要用户自己定义。但在通常情况下，用户并不需要把所有的输入或输出控件都与一个端口建立关联，与外部交换数据，因而需要改变连接端口中端口的个数。

在接线端口单击右键，弹出图 4.33 所示的快捷菜单，对接线端口样式进行设置。

LabVIEW 提供了两种方法来改变端口的个数。

(1) 在连接端口右键快捷菜单中选择“添加接线端”或“删除接





线端”命令，逐个添加或删除接线端口。这种方法较为灵活，但也比较麻烦。

(2) 在连线端口右键快捷菜单中选择“模式”命令，会出现一个图形化下拉菜单，菜单中会列出 36 种不同的连线端口，在一般情况下可以满足用户需求。这种方法较为简单，但是不够灵活，有时不能满足需要。

通常的做法是，先用第 2 种方法选择一个与实际需要比较接近的连线端口，然后再用第一种方法对选好的连接端口进行修正。

完成了连线端口的创建以后，下面的工作就是定义前面板中的输入和输出控件与连线端口中各输入输出端口的关联关系。

## 2. 连线端口

按照 LabVIEW 的定义，与输入控件相关联的连线端口作为输入端口。在子 VI 被其他 VI 调用时，只能从输入端口中输入数据，而不能从输入端口中向外输出数据。当某一个输入端口没有连接数据连线时，LabVIEW 就会将与该端口相关联的输入控件中的数据默认值作为该端口的数据输入值。相反，与输入控件相关联的连线端口都作为输出端口，只能向外输出数据，而不能向内输入数据。

(1) 接线端口位于前面板的右上角，图标位于前面板窗口及程序框图窗口的右上角，在连接端口的图标左侧。

(2) 将前面板置为当前面板，将鼠标放置在前面板右上角的连线端口图标上方，鼠标变为连线工具状态。单击鼠标右键，在弹出的快捷菜单中选择“模式”命令，同时在下一级菜单中显示接线端口模式，选择第一行第五个模式，如图 4.34 所示。

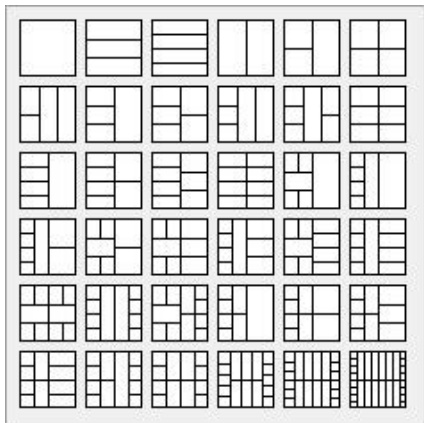


图 4.34 端口模式

(3) 将鼠标移动至连线板左侧上方的端口上，选中这个端口，端口变为黑色，如图 4.35 所示。

(4) 用鼠标在输入控件 A 上单击，选中输入控件 A，此时输入控件 A 的图标周围会出现一个虚线框，同时，黑色接线端口变为棕色。此时，这个端口就建立了与控件 A 的关联关系，端口的名称为 A，颜色为棕色，如图 4.36 所示。

(5) 使用同样的方法连接控件 B、C。端口的颜色是由与之关联的前面板对象的数据类型来确定的，不同的数据类型对应不同的颜色，例如，与布尔量相关联端口的颜色是绿色。

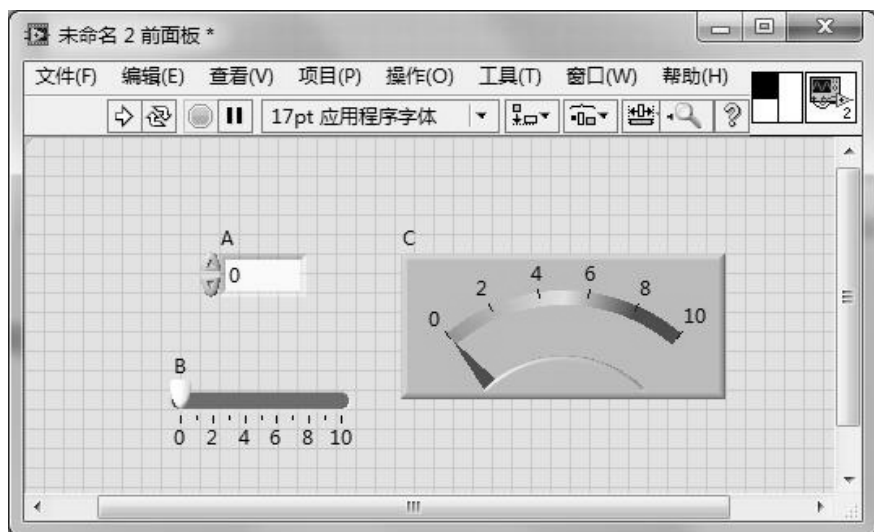


图 4.35 选中输入端口

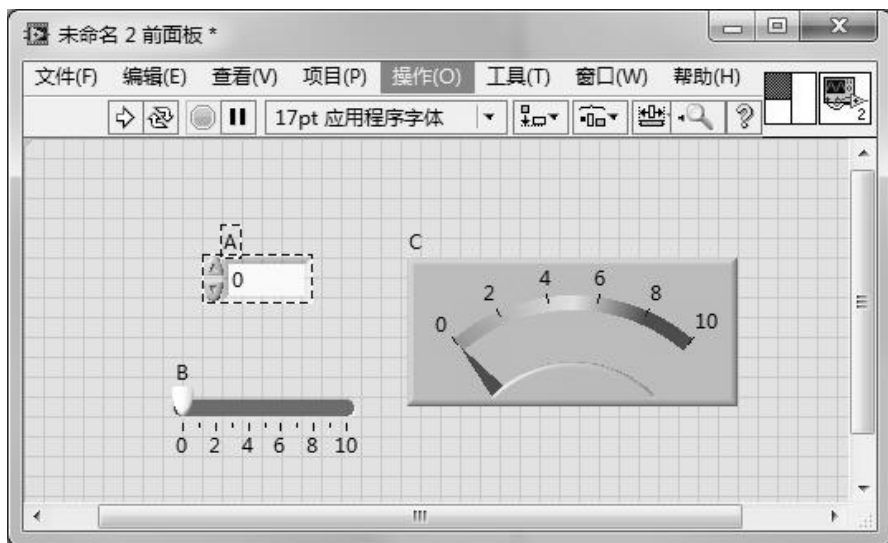


图 4.36 建立连线端口与输入控件 A 的关联关系

建立前面板中其他输入或输出控件与连线端口关系的方法与之相同。

在编辑调试 VI 过程中，用户有时会根据实际需要断开某些端口与前面板对象的关联。具体做法是在需要断开端口的右键快捷菜单中选择“断开连接本地接线端”。

若在快捷菜单中选择“断开连接全部接线端”命令，则会断开所有端口的关联。

### 3. 调用子 VI

在完成了连线端口的定义之后，这个 VI 就可以当作子 VI 来调用了。下面介绍如何在一个主 VI 中调用子 VI，具体步骤如下。

(1) 选择子 VI。在“函数”选板中选择“选择 VI”，如图 4.37 所示。



图 4.37 选择 VI

弹出“选择需要打开的 VI”对话框，如图 4.38 所示，找到需要调用的子 VI，选中后单击“打开”按钮。

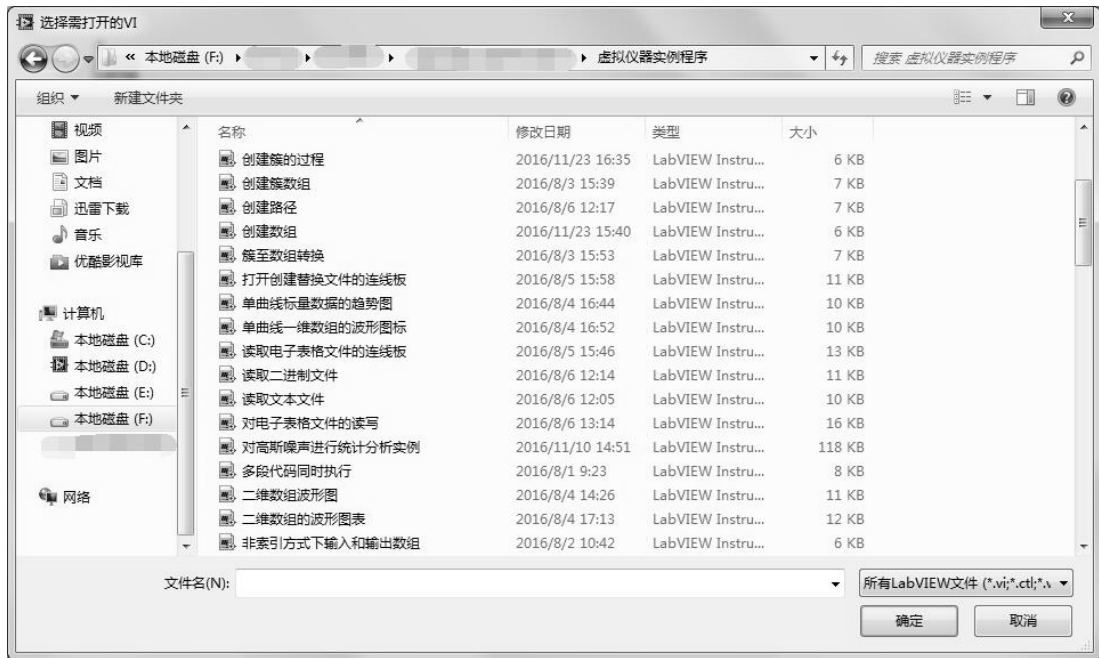


图 4.38 “选择需要打开的 VI”对话框

(2) 将子 VI 的图标放置在主 VI 程序框图窗口中。用户选择子 VI 后，在鼠标指针上会出现这个子 VI 的图标，将其移动到程序框图窗口的适当位置上，单击鼠标左键，将图标加入主 VI 的程序框图中。

(3) 用连线工具将子 VI 的各个连线端口与主 VI 的其他节点按照一定的逻辑关系连接起来。

至此，就完成了子 VI 的调用。

采用上述调用方式来调用子 VI，只是将其作为一般的计算模块来使用，程序运行时并不显示其前面板。如果需要将子 VI 的前面板作为弹出式对话框来使用，则需要改变一些 VI

的属性设置。

在子 VI 前面板窗口右上角图标的右键快捷菜单中选择“VI 属性”（或者在“文件”菜单中选择“VI 属性”），会出现一个“VI 属性”对话框，在对话框的“类别”下拉列表框中选择“窗口外观”，将对话框页面切换到窗口显示属性页面，如图 4.39 所示。



图 4.39 “VI 属性”对话框

在对话框中单击“自定义...”按钮，弹出“自定义窗口外观”对话框，如图 4.40 所示。在该对话框中选中“调用时显示前面板”和“如之前未打开则在运行后关闭”复选框，单击“确定”按钮关闭对话框。

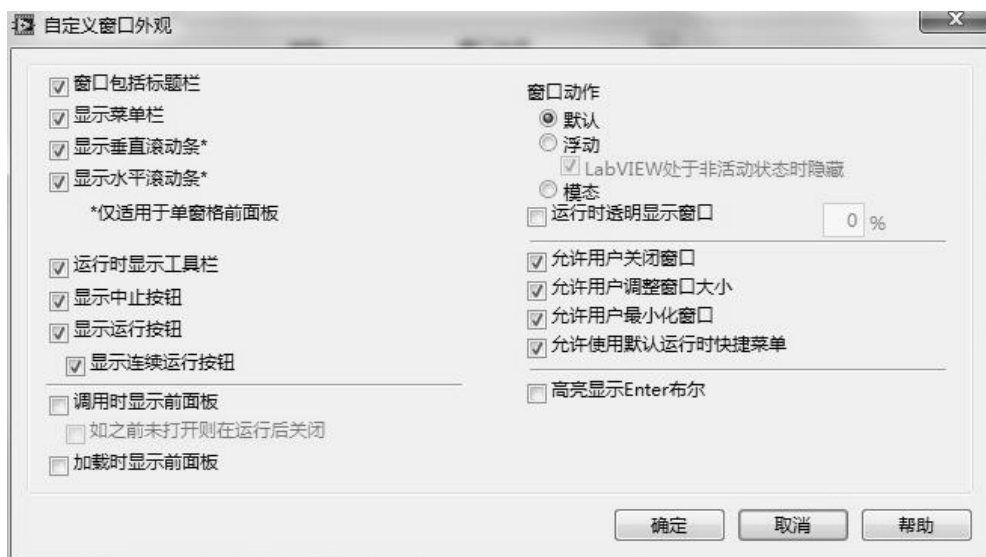


图 4.40 “自定义窗口外观”对话框

选中“调用时显示前面板”后，当程序运行到这个子 VI 时，其前面板就会自动弹出来。若再选中“如之前未打开则在运行后关闭”，则当子 VI 运行结束时，其前面板会自动消失。

## 4.6 性能和内存信息

性能和内存信息窗口是获取应用程序性能及内存使用情况的有力工具。性能和内存信息窗口采用交互式表格的形式，可以显示每个 VI 在系统中的运行时间及其内存使用的情况。表格中的每一行代表某个特定 VI 的信息，每个 VI 的运行时间被分类总结。性能和内存信息窗口可计算 VI 的最长、最短和平均运行时间。

通过本表格可以用交互的方式全部或部分显示和查看信息，将信息按类排序，或在调用某个特定 VI 的子 VI 时查看子 VI 运行性能的数据。

选择菜单栏中的“工具”→“性能分析”→“性能和内存”命令，可以显示性能和内存信息窗口，如图 4.41 所示。

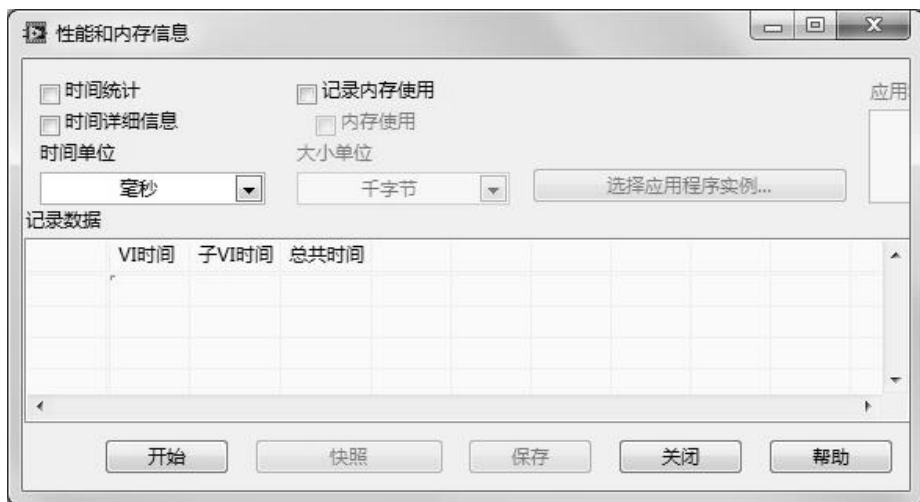


图 4.41 “性能和内存信息”窗口

收集内存使用信息将明显增加 VI 运行时间的系统开销，因此收集内存使用信息为可选操作。要在启动性能和内存信息窗口前正确勾选“记录内存使用”复选框以确认是否收集这部分数据。一旦记录会话开始，该复选框便无法更改。

可选择仅部分显示表格的信息。有些基本数据始终可见，但也可通过设置性能和内存信息窗口中的相关复选框来显示各种统计数据、详情和内存使用信息（被启用时）。另外，全局 VI 的性能信息也可显示。但这部分信息有时需要略有不同的解释，如下所述。

双击表格中的子 VI 名可查看子 VI 的性能数据。此时，在各 VI 的名称下将立即出现新的行，显示每个子 VI 的性能数据。上级全局 VI 的名称后，表格中将出现新的行，显示子面板上每个控件的性能数据。

单击某列列首可按想要的顺序排列表格中的各行数据。当前列排序的列首标题将以粗体显示。

VI 的计时并不一定与 VI 完成运行所需要的时间相对应。原因在于多线程执行系统可将两个或更多个 VI 的执行任务交错安排。另外，由于有一定数量的系统开销无法归于任何一个 VI，如用户响应对话框的时间，或程序框图中等待函数所占用的时间以及检查鼠标单击的时间等。

勾选时间统计复选框可查看关于 VI 即时的其他详细信息。

勾选时间详细信息复选框可查看将 VI 运行总时间进行细分后的计时类别。对于具有大量用户界面的 VI，这些类别可以帮助用户确定其中用时最多的操作。

勾选内存使用复选框可以查看 VI 对内存的使用情况。但该复选框仅在记录形成前勾选方可使用。所显示的数值表示了 VI 的数据空间对内存占用的程度，这部分数据空间不包括供支持所有 VI 使用的数据结构。VI 的数据空间不仅包含前面板控件所占用的显性数据空间，还包括编译器隐性创建的临时缓冲区所占用的数据空间。

VI 运行完毕后即可测得它所使用内存的大小，但可能无法反映其确切的使用总量。例如，如 VI 在运行过程中创建了庞大的数组，但在运行结束前数组有所减小，则最后显示的内存使用量便无法反映 VI 运行期间较大的内存使用量。

本部分显示两组数据：已使用的字节数和已使用的块数。块是一段用于保存单个数据的连续内存。例如，一个整数数组可以为多字节，但仅占用一个块。执行系统为数组、字符串、路径和图片使用独立的内存块。如应用程序内存中含有大量的块，将导致性能（不仅是执行性能）的整体下降。

## 4.7 搜索控件、VI 和函数

选择“查看”→“控件选板”或“函数选板”，打开控件或函数选板，选板顶部会出现两个按钮。

“搜索”按钮会将选板转换为搜索模式，基于文本查找选板上的控件、VI 或函数。选板处于搜索模式时，单击“返回”按钮可退出搜索模式，返回选板。

“自定义”按钮提供当前选板的模式选项、显示或隐藏所有选板的类别以及在文本和树形模式下按字母顺序对选板上各项进行排序。在快捷菜单中选择“选项”，可打开“选项”对话框中的控件/函数选板页，为所有选板选择显示模式。只有当单击选板左上角的图钉将选板锁住时，该按钮才会显示。

在熟悉 VI 和函数的位置之前，可以使用“搜索”按钮搜索函数或 VI。例如，需要查找“随机数”函数，可在函数选板工具条上单击“搜索”按钮，在选板顶部的文本框中输入随机数。LabVIEW 会列出所有匹配项，包括以输入文本作为起始的项和内容，包含输入文本的项，如图 4.42 所示，可以单击某个搜索结果并将其拖曳进入程序框图中。通过双击在选板上高亮显示搜索结果的位置。



图 4.42 搜索选板搜索“随机数”函数

# 第5章 数 据 操 作

## 5.1 概 述

作为一种通用编程语言，LabVIEW 与其他文本编辑语言一样，数据操作是最基本的操作。LabVIEW 支持几乎所有常用的数据类型和数据运算，同时还拥有其特殊的一些数据类型。本章主要介绍常用的数据类型：数字型、布尔型、枚举型、时间和日期类型、Variant 数据类型，以及局部变量和全局变量的使用。最后将介绍一些基本的数据运算，包括算术运算、关系运算、逻辑运算以及表达式节点。









## 5.2 数 据 类 型

数据结构是程序设计的基础，不同的数据类型和数据结构在 LabVIEW 中存储的方式是不一样的。选择合适的数据类型不但能提高程序的性能，还能节省内存的使用。LabVIEW 中接线端以不同的图标和颜色来表示不同的数据类型。

### 5.2.1 数字型

数字型是基本的数据类型，主要包括浮点型、整型和复数型 3 种类型。不同数据类型的差别在于存储数据使用的位数和值的范围。在此基础上还可以细分为更多具体类型，如表 5.1 所示。

表 5.1 数字型数据

输入控 件端子	显示控 件端子	数据类型	占用位数	取 值 范 围	默认值
		单精度浮点数	32	-3.40e+38~3.40e+38，绝对值最小可达 1.40e-45	0.0
		双精度浮点数	64	-1.79e+308~1.79e+308，绝对值最小可达 4.94e-324	0.0
		扩展精度浮点数	128	-1.19e+4932~1.19e+4932，绝对值最小可达 6.48e-4966	0.0
		单精度浮点复数	64	实部和虚部分别与单精度浮点数相同	0.0+i0.0



(续表)

输入控件端子	显示控件端子	数据类型	占用位数	取值范围	默认值
		双精度浮点复数	128	实部和虚部分别与双精度浮点数相同	0.0+i0.0
		扩展精度浮点复数	256	实部和虚部分别与扩展精度浮点数相同	0.0+i0.0
		定点数	64 或 72 (如包括上溢状态)	因用户配置而异	0.0
		8 位带符号整数	8	-128~127	0
		16 位带符号整数	16	-32 768~32 767	0
		32 位带符号整数	32	-2 147 483 648~2 147 483 647	0
		64 位带符号整数	64	-9223372036854775 808~-922337203685 4775807	0
		8 位无符号整数	8	0~255	0
		16 位无符号整数	16	0~65535	0
		32 位无符号整数	32	0~4294967295	0
		64 位无符号整数	64	0~18446744073709551615	0
		128 位时间标识	128	最小时间: 01/01/1600 00:00:00 UTC 最大时间: 01/01/3001 00:00:00 UTC	

LabVIEW 的数据类型隐含在前面板的输入控件和显示控件中。数值控件主要位于控件选板的“数值 (Numeric)”子选板中。

任何数据类型都具有相应的常数，常数在程序运行过程中是不可改变的。LabVIEW 中的常数只能在程序框图中使用，它存在于函数选板的“数值”子选板中。任何时候都可以右击控件或对象，选择“转换为常量 (Change to Constant)”选项，将控件转换为常数。

LabVIEW 中控件所代表的数据类型在编程时可以根据不同需要来改变。在前面板或程序框图中右击控件或对象，从右键快捷菜单中选择“表示法 (Representation)”选项打开选板，从中可以选择该控件所代表的数据类型。

**例 5-1** 定义数值输入控件的数据类型为单精度浮点型，最大值为 10，最小值为 0，默认值为 0，并设定增量的大小为 0.001，精度位数为 3。

(1) 启动 LabVIEW 2013，打开前面板，选择控件选板中的“新式→数值→数值输入控件”将其置于前面板中，右击数值输入控件，从右键快捷菜单中打开其属性对话框，切换到“数据类型”选项卡。单击“表示法”图标，选择单精度浮点型 SGL，如图 5.1 所示。

(2) 切换到“数据输入”选项卡，取消选中“使用默认界限”复选框，然后在“最小值”文本框中输入 0，在“最大值”文本框中输入 10，在“增量”文本框中输入 0.001，如图 5.2 所示。

(3) 选择“显示格式”选项卡，将“精度类型”选择为“精度位数”，将“位数”设置为 3，并选中“隐藏无效零”复选框，使控件能显示数值后面的无效零，如图 5.3 所示。



图 5.1 “数值属性：数值”对话框



图 5.2 定义数据输入

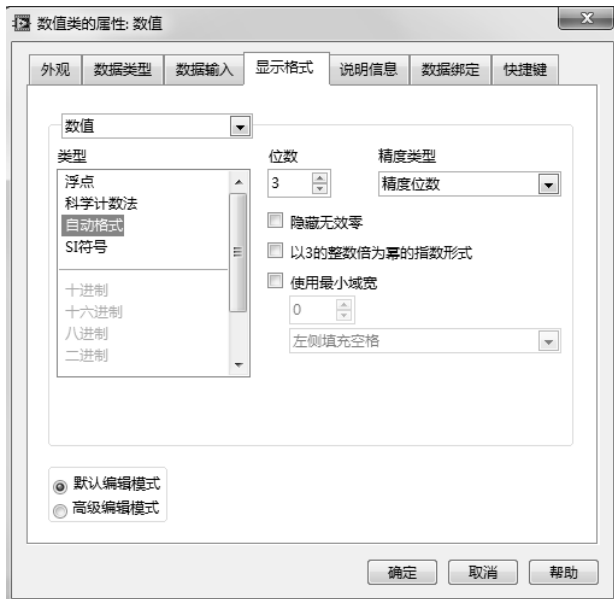


图 5.3 设置格式与精度

(4) 设置完成后单击“确定”按钮，此时查看前面板的数值输入控件，发现其默认值已变成 0.000，单击控件左边的增量按钮，发现数值按 0.001 递增。

### 5.2.2 布尔型

布尔型比较简单，只有 1 和 0，或真（True）和假（False）两种状态，也称逻辑型。布尔型主要包含在控件选板的“布尔（Boolean）”子选板中，如图 5.4 所示。和数字型类似，布尔常数存在于函数选板的“布尔（Boolean）”子选板中，如图 5.5 所示。



图 5.4 “布尔”子选板



图 5.5 布尔常数

布尔型输入控件的一个重要属性是机械动作 (Mechanical Action)，使用该属性可以模拟真实开关的动作特性。右击布尔型输入控件，选择“机械动作 (Mechanical Action)”选项就可以打开子菜单。

在布尔输入控件属性对话框的“属性 (Operation)”标签中也可以设置机械动作，而且还有详细的说明和动作效果预览。

### 5.2.3 枚举类型

LabVIEW 中的枚举类型和 C 语言中的枚举类型定义相同。它提供了一个选项列表，其中每一项都包含一个字符串标识和数字标识，数字标识与每一选项在列表中的顺序一一对应。枚举类型包含在控件选板的“下拉列表与枚举 (Ring&Enum)”子选板中，如图 5.6 所示，而枚举常量包含在函数选板的“数值 (Numeric)”子选板中，如图 5.7 所示。

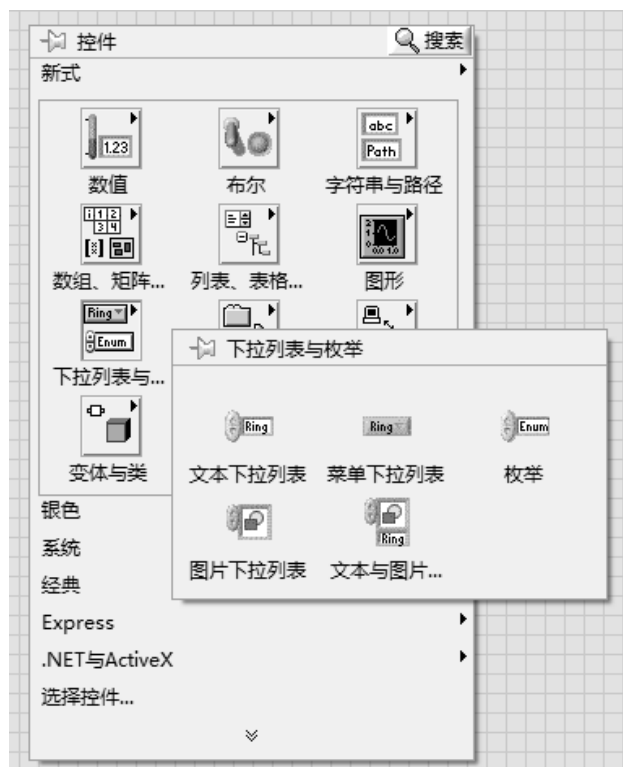


图 5.6 枚举类型



图 5.7 枚举常量

枚举数据类型可以用 8 位、16 位或 32 位无符号整数表示，这 3 种表示方式之间的转换可以通过右键快捷菜单中的“表示法 (Representation)”选项实现。

任何时候，枚举控件的值一定是  $0 \sim n-1$  的正整数 ( $n$  为枚举项目总数)，而“菜单下拉列表 (Ring)”控件的数值则可以任意。

### 例 5-2 实现加、减、乘、除四则运算。

(1) 启动 LabVIEW2013，打开前面板，选择控件选板中的“新式→数值→数值输入控件”，将两个“数值输入控件”置于前面板中，分别命名为“x”和“y”；选择“控件选板→数值→数值输出控件”置于前面板中，命名为“结果”。

(2) 选择“新式→下拉表与枚举→枚举”控件置于前面板中，命名为“运算”，右击该控件，从弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框中选择“编辑项”选项卡设置枚举值，如图 5.8 所示。

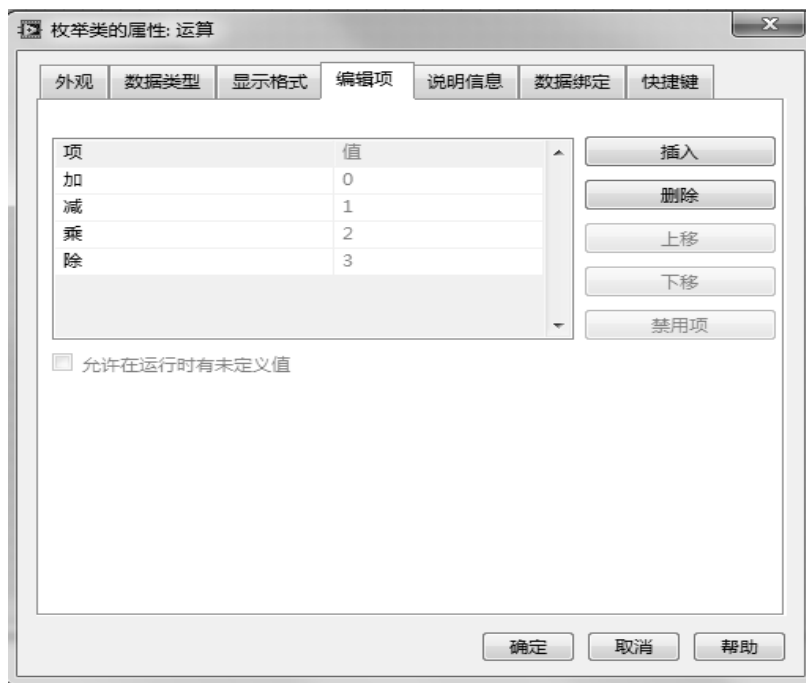


图 5.8 枚举类属性对话框

(3) 切换到程序框图，选择函数选板中的“编程→结构→条件结构”控件放置在程序框图中，将“运算”控件连接到条件结构控件的条件输入端，此时第一个分支“加”即成为“默认分支”。选择“编程→数值→加”控件放置在默认“加”分支内，输入端口分别与“x”、“y”连接，输出端口与“结果”连接。“加”分支结构如图 5.9 所示。

(4) 单击条件结构选择器两侧的黑头，切换到第二分支，用同样的方法添加“减”运算，输出端口与“结果”相连。在条件结构体边框上右击，从弹出的快捷菜单中选择“在后面添加分支”命令，此时“选择条件器”中自动显示“乘”运算，在此分支中添加“乘”控件，输出端口与“结果”相连。用同样的方法添加“除”运算分支，输出端口与“结果”相连。在前面板上对 x、y 进行赋值，运行程序，前面板和程序框图如图 5.10 所示。

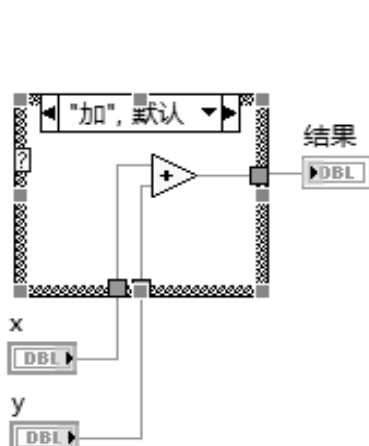


图 5.9 “加”分支结构

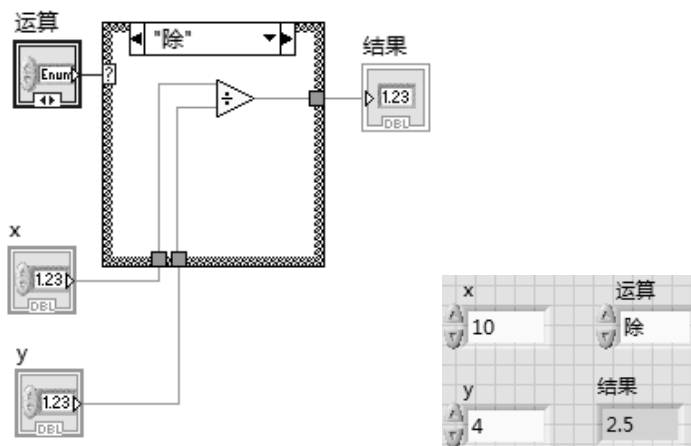


图 5.10 前面板和程序框图四则运算

### 5.2.4 时间类型 (Time Stamp)

时间类型是 LabVIEW 中特有的数据类型，用于输入与输出时间和日期。时间标识控件位于控件选板的“数值 (Numeric)”子选板中，如图 5.11 所示。相应常数位于函数选板的“定时 (Timing)”子选板中，如图 5.12 所示。

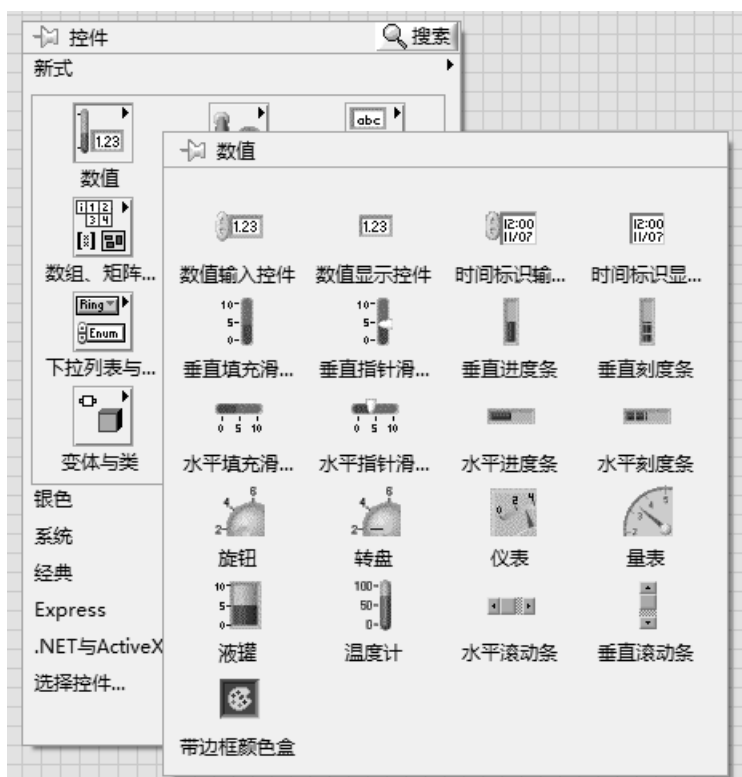


图 5.11 时间标识控件

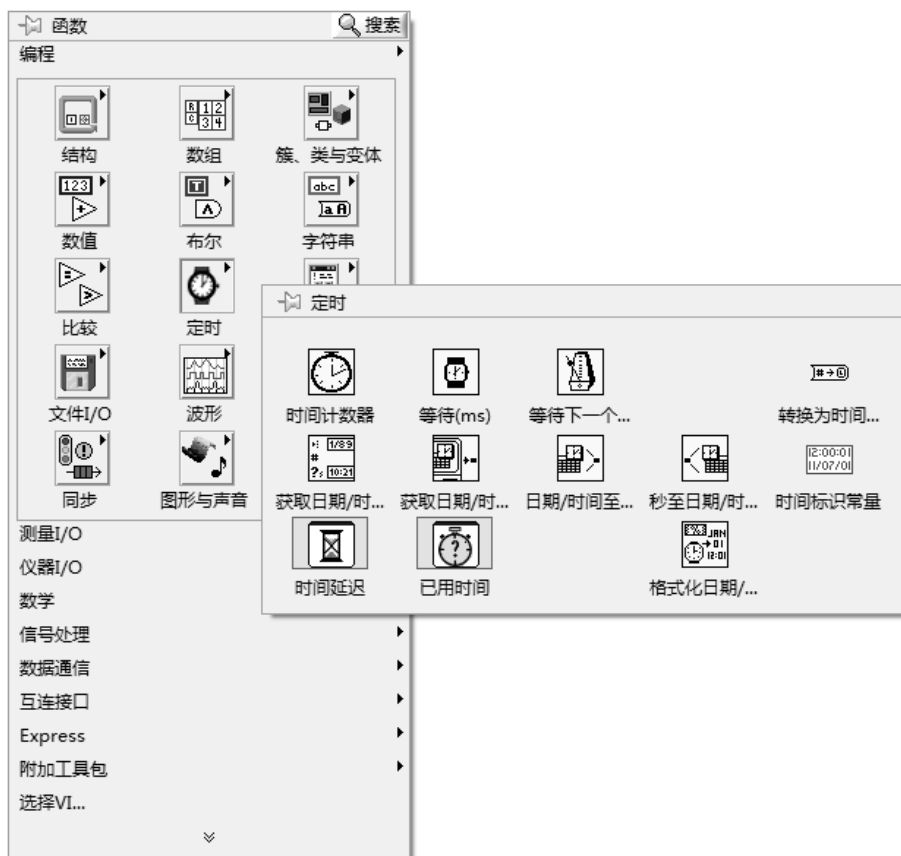


图 5.12 时间类型常数

右击时间标识控件，选择“显示格式（Format&Precision）”选项，可以设置时间日期的显示格式和显示精度。单击时间日期控件旁边的“时间与日期选择”按钮，可以打开时间和日期选择对话框，在此对话框中可以选择时间和日期。

### 5.2.5 变体数据类型（Variant）

变体数据类型和其他的数据类型不同，它不仅能存储控件的名称和数据，而且还能携带控件的属性。例如，当要把一个字符串转换为变体数据类型时，它既保存字符串文本，而且还标识这个文本为字符串类型。LabVIEW 中的任何一种数据类型都可以使用相应的函数来转换为变体数据类型。该数据类型包含在前面板控件选板的“变体与类(Variant&Class)”子选板中，如图 5.13 所示。

“变体”数据类型主要用在 ActiveX 技术中，以方便不同程序之间的数据交互。在 LabVIEW 中可以把任何数据都转换为“变体”数据类型。这样可以仅使用一种数据类型与其他程序通信，从而简化通信接口。因此可以称之为“通用”数据类型。



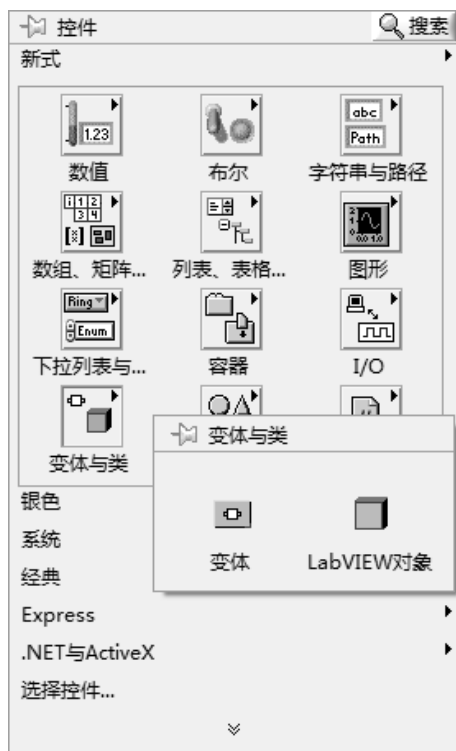


图 5.13 “变体与类”子选板

### 5.2.6 局部变量和全局变量

由于 LabVIEW 图形化编辑的特点，在有些情况下要在同一 VI 的不同位置或在不同的 VI 中访问同一个控件，这时控件对象之间的连线就无法实现。这时就需要用到局部变量或全局变量，通过局部变量或全局变量可以在程序框图中的多个地方读写同一个控件。局部变量只能用于同一个 VI 的程序框图，而全局变量可以用于同一台计算机中的多个 VI。

#### 1) 局部变量

局部变量只是在同一程序内部使用，每个局部变量都对应前面板上的一个控件，一个控件可以创建多个局部变量。局部变量位于函数选板的“结构 (Structures)”子选板中，如图 5.14 所示。

局部变量的创建有两种方法。第一种是从函数选板的“结构”子选板中选中“局部变量 (Local Variable)”节点，将其添加到程序框图中，这时由于局部变量还没有和其相应的输入或显示控件相关联，故图标上显示一个问号。单击此图标显示一个下拉菜单，其中列出了前面板上的控件名称，选择其中的名称就完成一个局部变量的创建，或右击此图标，从快捷菜单中选择“选择项 (Select Item)”选项也可以完成局部变量的创建。

第二种创建局部变量的方法是在前面板或程序框图中右击需要创建局部变量的控件，选择“创建→局部变量 (Create|Local Variable)”选项创建该控件的局部变量。



图 5.14 局部变量

## 2) 全局变量

LabVIEW 中的全局变量是以独立的 VI 文件形式存在的，这个 VI 文件只有前面板，没有程序框图，不能进行编程。通过全局变量可以在不同的 VI 之间进行数据交换，一个全局变量的 VI 文件可以包含多个不同数据类型的全局变量。与局部变量一样，全局变量也位于函数选板的“结构 (Structures)”子选板中，如图 5.15 所示。

全局变量的创建与局部变量不同，全局变量以一个独立的 VI 存在。从函数选板的“结构”子选板中选中“全局变量 (Global Variable)”节点，将其添加到程序框图中，然后双击全局变量图标打开其前面板，并向其中添加所需数据对应的控件，控件的类型和数量是没有限制的；最后保存该全局变量为一个独立的 VI。

创建并保存全局变量 VI 后，单击程序框图中全局变量的图标或右击该图标，从快捷菜单中选择“选择项 (Select Item)”选项，在列表中选择所包含的对象名称即可完成全局变量的选择。

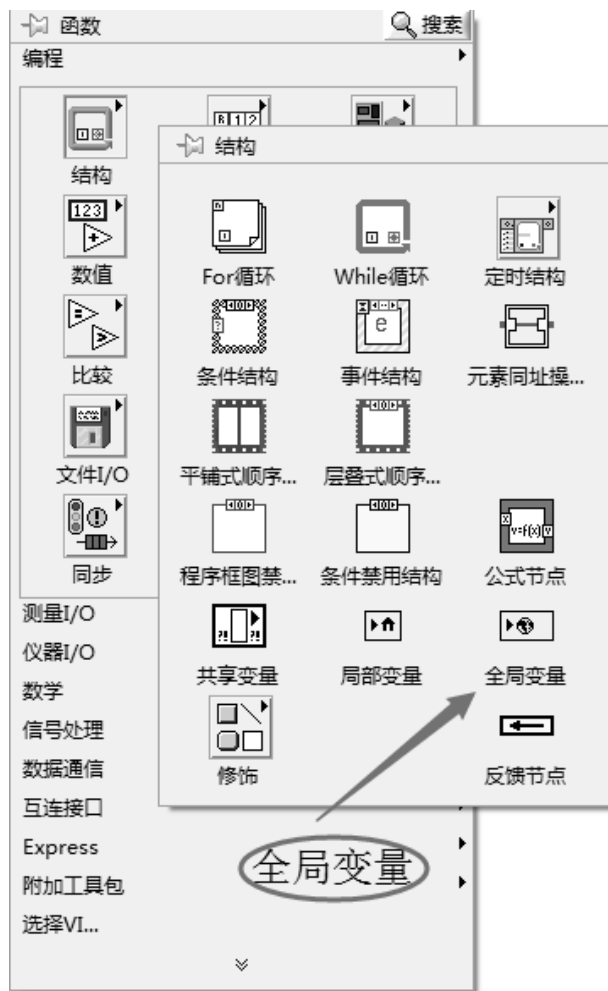


图 5.15 全局变量

## 5.3 数据运算

LabVIEW 中提供了丰富的数据运算功能，除了基本的数据运算外，还有许多功能强大的函数节点，并且还支持通过一些简单的文本脚本进行数据运算。与文本语言编辑不同的是，在文本语言编程中都具有运算优先级和结合性的概念，而 LabVIEW 是图形化编程，不具有这些概念，运算是按照从左到右沿数据流的方向顺序执行的。本节只介绍一些基本的数据运算方法，其他方法在后续的不同章节中介绍，或直接参考联机帮助文档。

### 5.3.1 算术运算

基本算术运算包含在函数选板的“数值 (Numeric)”子选板中，该子选板中有类型转换

(Conversion) 节点、复数 (Complex) 节点、数学和科学常数 (Math&Scientific Constants) 节点等, 如图 5.16 所示。



图 5.16 算术运算

基本算术运算主要实现加、减、乘、除等功能, LabVIEW 中的算术运算输入端能根据输入数据类型的不同自动匹配, 并且还能自动进行强制数据类型转换。

### 5.3.2 关系运算

关系运算也叫比较运算, 包含在函数选板的“比较”子选板中, 如图 5.17 所示。

使用关系运算可以进行数值比较、布尔值比较、字符串比较、数组比较和簇比较。不同类型的数据进行比较时, 规则是不同的, 简单介绍如下。

#### 1) 数值比较

数值比较是相同类型数据的比较, 数据类型不同时, 关系运算符输入端能自动进行强制数据类型转换, 然后再比较。



图 5.17 关系运算

## 2) 布尔值比较

布尔值比较实际上就是 0 和 1 两个值的比较，True 值大于 False 值。

## 3) 字符串比较

首先先说一下字符的比较，当两个字符比较时是按其 ASCII 码的大小来比较的。两个字符串的比较就是从字符串的第一个字符开始逐个比较字符，直到有两个字符不相等为止。

## 4) 数组比较和簇比较

数组的比较、簇的比较与字符串的比较类似，是从数组或簇的第 0 个元素开始比较，直到有不相等的元素为止。进行簇的比较时，簇中的元素个数、元素的数据类型和顺序必须相同。

### 5.3.3 逻辑运算

逻辑运算又称布尔运算，包含在程序框图函数选板的“布尔（Boolean）”子选板中，如图 5.18 所示。

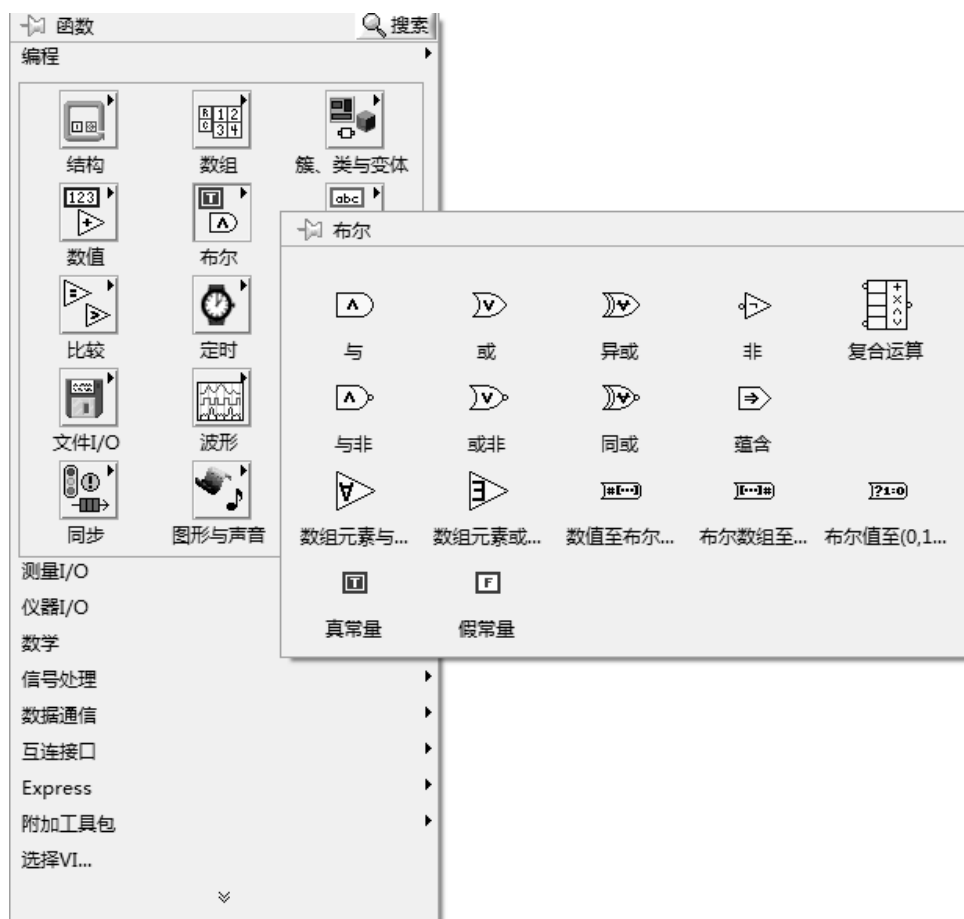


图 5.18 逻辑运算

LabVIEW 中逻辑运算的图标与数字电路中逻辑运算的图标相似。

逻辑运算符的输入数据类型可以是布尔型、整型、元素为布尔型或整型的数组和簇。当输入数据类型为整型时，运算符自动将整型数转换为相应的二进制数，然后再将转换后的二进制数每一位进行逻辑运算，最后的输出结果是经过逻辑运算后的十进制数。如果输入的数据类型是浮点型，运算符自动将它强制转换为整型数后再运算。

### 5.3.4 表达式节点

表达式节点包含在函数选板的“数值（Numeric）”子选板中，如图 5.19 所示。



图 5.19 表达式节点

使用表达式节点可以计算包含一个变量的数学表达式，该节点允许使用除复数外的任何数字类型。在表达式节点中可以使用的函数有 `abs`、`acos`、`acosh`、`asin`、`asinh`、`atan`、`atanh`、`ceil`、`cos`、`cosh`、`cot`、`csc`、`exp`、`expml`、`floor`、`getexp`、`getman`、`int`、`intrz`、`ln`、`lnpl`、`log`、`log2`、`max`、`min`、`mod`、`rand`、`rem`、`sec`、`sign`、`sin`、`sinc`、`sinh`、`sqrt`、`tan`、`tanh`。

在表达式节点可以使用许多运算符，其中有很多是双目和三目运算符，由于表达式节点是单变量输入的节点，所以在双目和三目运算符中除输入变量外只能使用常数。

**例 5-3** 创建一个 VI，给 VI 程序实现的功能为通过比较两个数值的大小，输出较大值的数的 3 倍和两数的差值。

(1) 首先启动 LabVIEW2013，在启动的初始化界面中选择“文件→新建 VI”新建一个空白 VI；也可选择“创建项目→VI”或者用快捷键“Ctrl+N”创建一个新的 VI。

(2) 在前面板上选择“控件选板→数值→数值输入控件”，添加两个数值输入控件，用工具选板中的“标签工具”或左键双击控件标签，分别修改标签为“x”和“y”；选择“控件选板→数值→数值输出控件”，添加两个数值显示控件，修改标签为“较大值的 3 倍”和“差值”，如图 5.20 所示。

(3) 在前面板窗口中通过“窗口→显示程序框图”或用快捷键“Ctrl+E”切换到程序框图。此时前面板中的控件以接线端的形式出现在程序框图中。

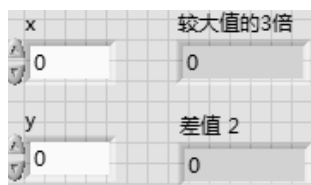


图 5.20 创建前面板

选择函数选板的“编程→比较→选择”、“编程→比较→大于”、“编程→数值→乘”和“编程→数值→减”控件置于程序框图中。利用连接工具将接线端和节点对象连接起来，如图 5.21 所示。

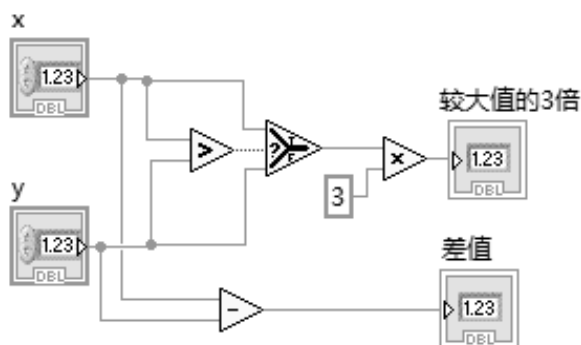


图 5.21 程序框图

切换到前面板中，设置数值输入参数，运行程序，观察结果，如图 2.22 所示。



图 5.22 运行结果

**例 5-4** 建立一个子 VI，具有勾股定理的功能（要求：输入两边长，即可输出第三边长）。

(1) 启动 LabVIEW 2013，在启动的初始化界面中选择“文件→新建 VI”，新建一个空白 VI，在前面板的控件选板中选择“编程→数值→数值输入控件”，将两个数值输入控件放置在前面板，左键双击控件标签，分别修改标签为“直角边 a、直角边 b”，单击鼠标右键，在弹出的快捷菜单中选择项目可以对控件的参数进行配置。用同样的方法创建显示控件，标签改为“斜边 c”。

(2) 在前面板窗口中通过“窗口→显示程序框图”或用快捷键“Ctrl+E”切换到程序框图。在程序框图窗口的函数选板中，依次选择“编程→数值→平方”、“编程→数值→加法”和“编程→数值→开平方”函数，用连线工具将其连接。VI 程序框图如图 5.23 所示。

(3) 在 VI 前面板右上角的 VI 图标上单击鼠标右键，从弹出的快捷菜单中选择“图标编辑”命令，进入图标编辑器窗口，通过“编辑→清空所有”命令清除所有的图形，在空白区



编辑图标：在图标编辑器中可以用“图标文本”或“符合”工具总的各类元素编辑图标，创建的图标如图 5.24 所示。

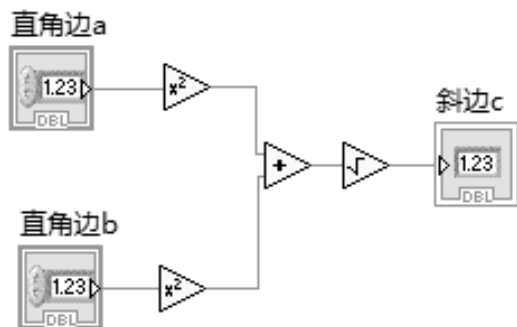



图 5.23 VI 程序框图

$$\sqrt{a^2 + b^2} = c$$

图 5.24 子 VI 图标

(4) 定义连接端子。连接端子用于指示 VI 的数据输入端和输出端。右击前面板右上角图标，从弹出的快捷键菜单中选择“模式命令”，选择 ，单击左侧上方格，然后在前面板中单击输入控件“直角边 a”，图标变为蓝色，说明该图标端子与“直角边 a”输入控件相连，用同样的方法连接左侧下方格与“直角边 b”、右侧方格与“斜边 c”显示控件。

(5) 将 VI 命名为“勾股定理”，保存程序后就可以在其他 VI 中被调用。

在一个 VI 程序中调用一个子 VI 的步骤是：在程序框图选择“查看→函数选板→选择 VI”命令，找到子 VI 保存的路径，打开该子 VI，此时在程序框图窗口中会显示已编辑好的子 VI 图标。

# 第6章 程序结构

任何计算机语言都离不开程序结构，LabVIEW 作为一种图形化的高级程序开发语言也不例外。除了 goto 语句，所有 C 语言中的程序结构都能在 LabVIEW 中找到对应的实现方法。此外，LabVIEW 中还有一些独特的程序结构，例如，事件结构、使能结构、公式节点和数学脚本节点等，因此通过 LabVIEW 可以非常方便快速地实现任何复杂的程序结构。

由于 LabVIEW 是图形化编程语言，它的代码以图形化的形式表现，因此无论是循环结构、Case 结构还是公式节点，它们都表现为一个方框包围起来的图形代码，这个方框就类似于 C 语言中的两个花括号 {}。

## 6.1 顺序结构

### 1. LabVIEW 程序的执行顺序

如果用传统的文本程序语言写过程序，一定想当然地认为程序是按照语言的顺序从上到下逐条执行的，除非遇到 goto 语句或者函数才会跳到另外一段代码执行。但是作为一种图形化开发语言，LabVIEW 有它独特的方法来确定其执行顺序。首先是根据数据流执行，只有当节点的所有输入点的数据都“流到”时，才会执行该节点。一般来说数据都是按照从左到右的方向“流动”的，这段代码就代表了结果 = (5 + 随机数) × 2 这个表达式，如图 6.1 所示。

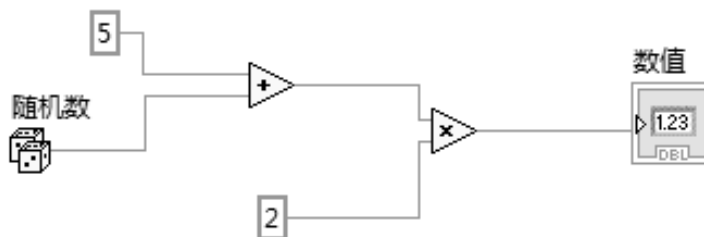


图 6.1 数据流式的编程方法

如果再在同一个框图（Block Diagram）内放一段类似代码的话，这两段代码并不是按照从左到右或从上到下的顺序执行的，而是并行独立执行的，如图 6.2 所示。这是因为 LabVIEW 实现了自动多线程，它使代码的执行效率大大提高了。如果用文本程序语言，实现多线程编程是非常费力的。

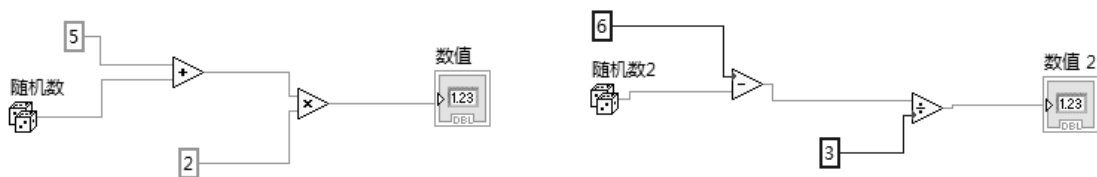


图 6.2 多段代码同时执行

## 2. 平铺式顺序结构和层叠式顺序结构

按照上述办法虽然能提高代码执行效率，但用户不知道哪一段代码是先执行的。在很多情况下，程序员也需要多端代码能按照设定的顺序执行。这时候就可用顺序结构（Sequence Structure）来帮忙了。有两种顺序结构，一种是平铺式顺序结构（Flat Sequence Structure），它是按照从左到右的顺序执行的。这样的好处是用户同时能看到所有的代码，但是当代码段数太多时就很难看了。另一种是层叠式顺序结构（Stacked Sequence Structure），它按照标定的顺序执行代码。与平铺式顺序结构不同的是，它将每段代码都“叠放”在了一起，因此用户同时只能看到一段代码。

平铺式顺序结构和层叠式顺序结构在函数选板中的位置，如图 6.3 所示。

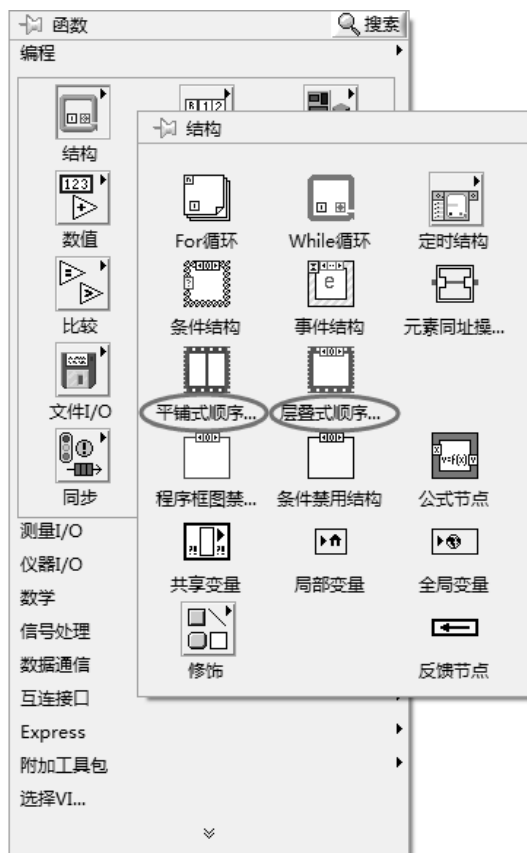


图 6.3 顺序结构在函数选板中的位置

当单击函数选板中对应的图标后，再在程序框图中按住鼠标左键向右下方拉动，估计大小合适的时候放开鼠标左键即可。可以看出它很像电影胶片，意思就是一帧一帧按顺序执行。通过右击其边框选择在“前面添加帧（Add Frame Before）”或“在后面添加帧（Add Frame After）”选项来增加帧。同理，可以选择“删除本帧（Delete This Frame）”选项来删除帧。将图 6.2 中的两段代码分别改写为按照平铺式顺序结构和层叠式顺序结构执行的两段代码，如图 6.4 和图 6.5 所示。

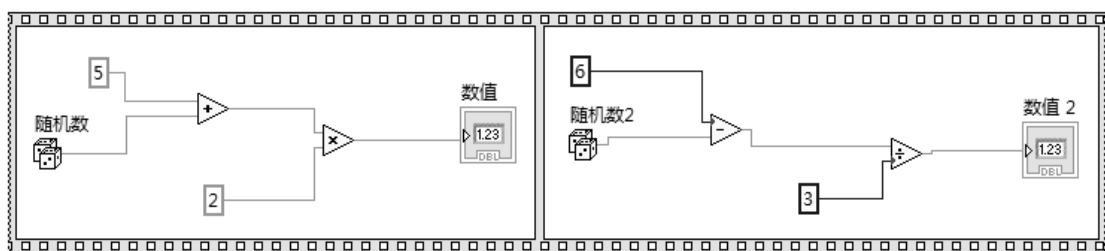


图 6.4 平铺式顺序结构

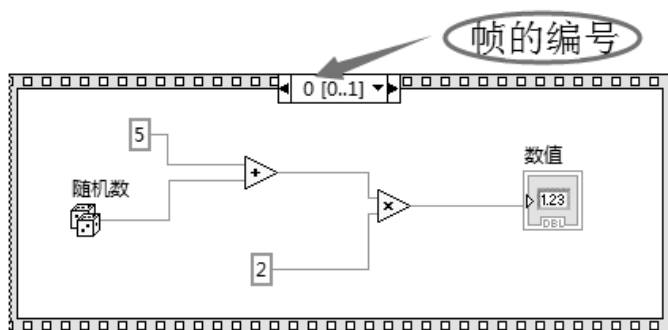


图 6.5 层叠式顺序结构

### 3. 在帧间传递数据

对于平铺式顺序结构，只需要将帧间的数据流用线连起来，就能实现将前一个帧的数据传到后一个帧中去，如图 6.6 所示。其中帧之间的数据连接点是可以鼠标拖动的，用户可以通过拖动来合理布线，到各种程序结构的数据连接端子都是同样可以拖动的。

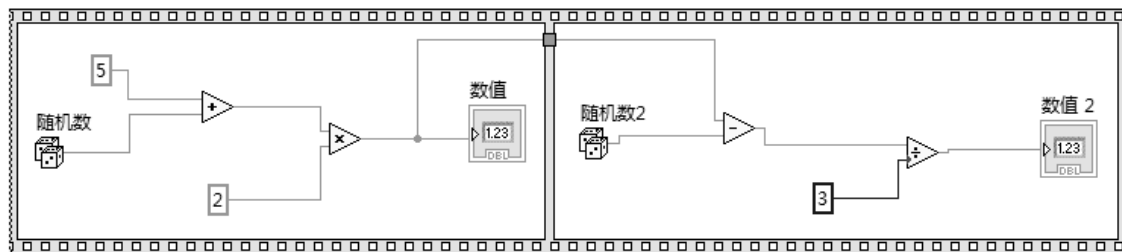


图 6.6 平铺式顺序结构在帧间传递数据

而对于层叠式顺序结构则要复杂一些。首先右击前一帧的下边框，选择“添加顺序局部

变量 (Add Sequence Local)”选项，于是右击之处就出现了一个小黄色端子，可以将其拖到边框的任何位置，然后将需要传递的数据与该端子连接。接着进入后一帧，这时仍然能看到这个端子，将其与需要连接的点连接即可，如图 6.7 所示。

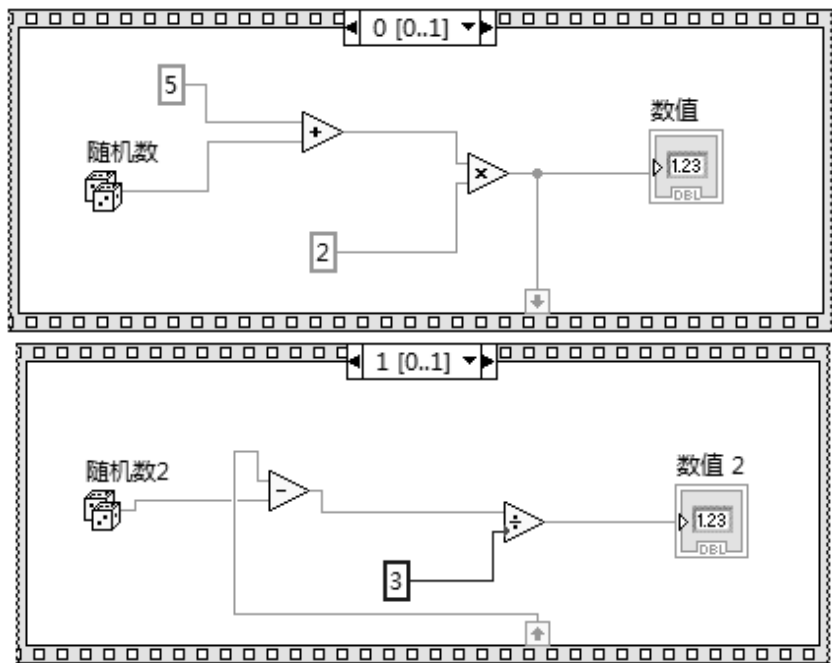


图 6.7 层叠式顺序结构在帧间传递数据

## 6.2 循环结构

LabVIEW 中的循环结构有 For 循环、While 循环和定时循环 3 种，其功能与文本编程语言的循环结构类似，可以控制循环体内的代码执行多次。

### 6.2.1 For 循环

#### 1. 初识 For 循环

For 循环用于将某段程序循环执行指定的次数。可以通过两种方法指定循环次数，一种是直接给定，一种是通过输入数组的大小给定。它在函数选板中的位置如图 6.8 所示。

当单击功能面板中对应的图标后，再在程序框图中按住鼠标左键向右下方拉动，估计大小合适的时候放开鼠标左键即可。下面写一段最简单的代码将当前循环次数的值赋给控件 a。等效的伪代码如下：



图 6.8 For 循环在函数选板中的位置

For i=0 to N-1

{a=i;}

End

如图 6.9 所示，其中 N 代表循环次数，i 代表当前是第几个循环。

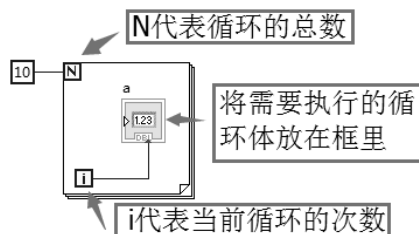


图 6.9 最简单的 For 循环

若需要与循环体外部交换数据，只需要将数据点用线连接起来即可，如图 6.10 所示。当然也可以采用局部变量或全局变量的方式。

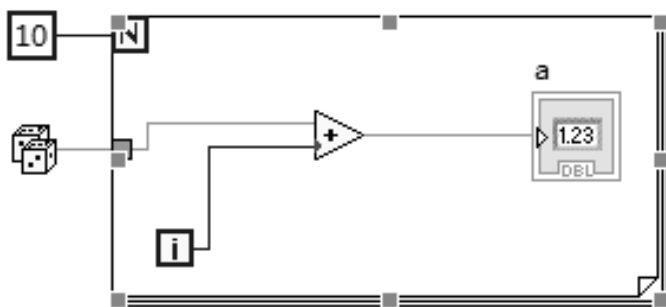


图 6.10 与循环体外部代码交换数据

## 2. 输入和输出数组

如果输入 For 循环体的数据为数组的话，有两种情况：一种是将整个数组一次性输入循环体；另一种是让数组元素逐个输入，并以数组大小确定循环次数。同理，输出也一样，可以一次性整个数组输出，也可以将元素逐个输出，最后再将所有输出元素组合为同一个数组输出。一般来说，如果直接将数组与内部数据连接，默认就是自动索引，即数组元素一个个地输入，输出也是一样的。在一般情况下，用户不再需要指定  $N$ ，即运行的次数，它会自动根据数组大小执行。当然如果用户硬要指定一个  $N$ ，那么它按照最小的执行次数执行。即如果数组有 5 个元素，而用户指定  $N$  为 10，则实际循环次数为 5。索引方式下输入和输出的一维数组如图 6.11 所示。

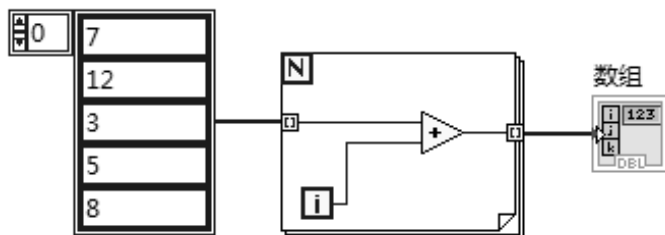


图 6.11 索引方式下输入和输出一维数组

如果需要将数组一次性完整输入，则需要右击输入点并选择“禁用索引 (Disable Indexing)”选项关闭索引，同理输出也一样，如图 6.12 所示。这样的话，数组将一次性输入 For 循环，而不是逐个输入。

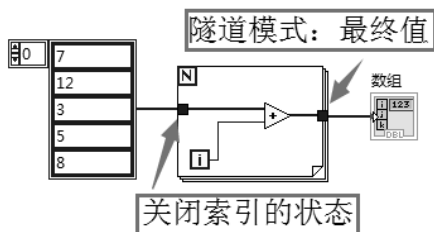


图 6.12 非索引方式下输入和输出数组

对于二维数组或多维数组，方法也是一样的。如果采取索引的办法输入一个二维数组，则最外层循环按行输入，内层循环按照输入行的元素逐个输入，如图 6.13 所示。多维数组依此类推。

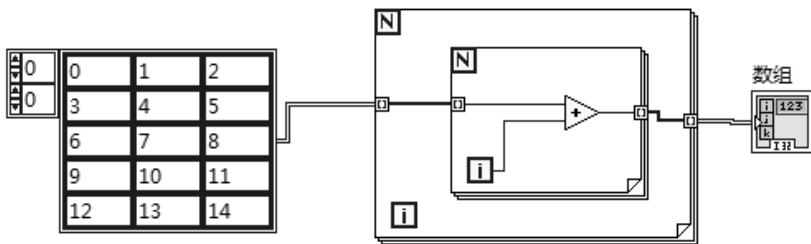


图 6.13 索引方式下二维数组的输入和输出

### 3. 移位寄存器 (Shift Register)

移位寄存器，顾名思义，就是把上一次循环产生的结果“移动”到下一次循环的输入。添加移位寄存器的方法是右击 For 循环的边框，选择“添加移位寄存器 (Add Shift Register)”选项，在右击点和另一侧就会出现一个小黑框，把它与需要连接的数据连接即可。

### 4. 反馈节点 (Feedback Node)

除了使用移位寄存器能实现前后两次循环之间的数据交换，LabVIEW 还提供了反馈节点来实现这个功能。反馈节点在功能上完全等价于移位寄存器。但使用反馈节点可以让程序更简洁，更容易读懂。使用起来也非常简单，就是直接将输入与输出连接起来，这时就会自动出现一个反馈节点的符号。当然，同移位寄存器一样，最好指定反馈节点的初值。

**例 6-1** 判断一个大于 1 的整数  $m$  是否为素数。

(1) 启动 LabVIEW 2013，打开前面板，选择“控件选板→数值→数值输入控件”，将其置于前面板中，并命名为“ $m$ ”。

(2) 打开程序框图，选择“函数选板→结构→平铺式”控件，将其置于程序框图中；选择“函数选板→结构→For 循环”控件，将其置于顺序结构内；选择“函数选板→数值→平方根”和“函数选板→数值→最近数取整”控件，将它们置于顺序结构内、For 循环体外，并进行连线，如图 6.14 所示。

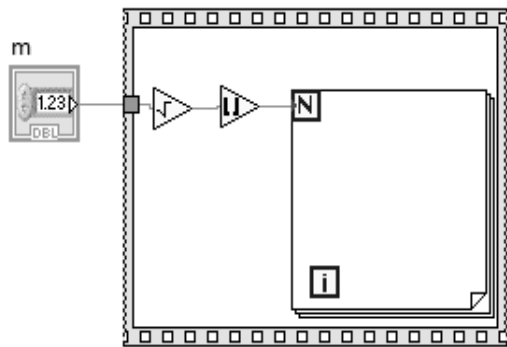


图 6.14 确定循环结构的循环次数



(3) 选择“函数选板→数值→加”和“函数选板→数值→商与余数”控件，将其分别置于“For 循环”结构中，“加”的一个输入端口与“For 循环”结构的“循环计数  $i$ ”连接，另一个输入端口通过右键创建一个常量 2，输出端口与“商与余数”的一个输入端连接，构建  $m$  被  $2 \sim \sqrt{m}$  之间任一整数除的程序如图 6.15 所示。

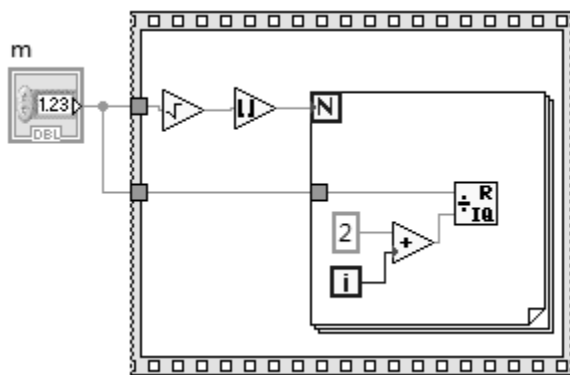


图 6.15 构建  $m$  被  $2 \sim \sqrt{m}$  之间任一整数除的程序

(4) 选择“函数选板→结构→条件结构”控件，将其置于 For 循环体内，将“商与余数”节点的余数端口连接到条件结构的选择端口上。

(5) 在 For 循环结构的边框上单击鼠标右键，在弹出的快捷菜单中选择“条件接线端”命令，然后在 For 循环结构内设置一个条件接线端，该条件接线端默认为“真 (T) 时停止”。选择“函数选板→布尔→真常量”，在条件结构的 0 分支程序框内添加一个布“真常量”，并将其与条件结构外的 For 循环条件接线端相连，则当余数为 0 时就会停止。

(6) 单击条件结构选择器两侧的黑头，切换到条件结构 1 分支程序，选择“函数选板→布尔→假常量”，在此分支中添加一个“假常量”，同时将其与 For 循环条件接线端相连。在分支 1 程序中将  $i+2$  的值加上 1 后输出到输出通道上，分支 0 程序将  $i+2$  的值直接输出到输出通道上，此时的程序框图如图 6.16 所示。

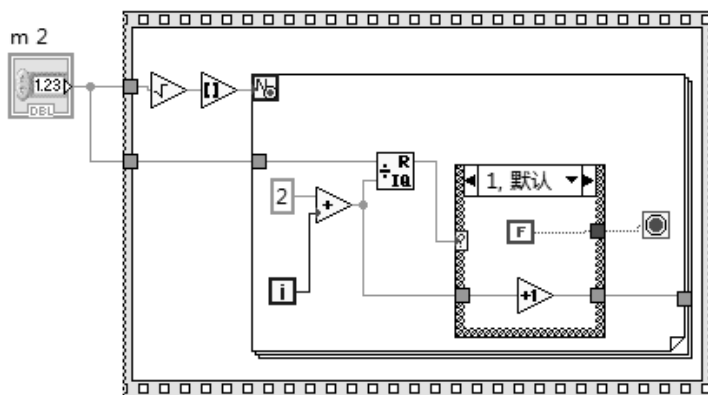


图 6.16 判断是否整除

(7) 在顺序结构的右侧边框上单击鼠标右键，在弹出的快捷菜单中选择“在后面板添加帧”命令，添加一帧。

(8) 选择“函数选板→比较→大于”节点，将其置于顺序结构的第 2 帧内，并将 For 循环输出通道的输出值与  $\sqrt{m}$  分别接到“大于”节点的两个输入端口。选择“函数选板→结构→条件结构”控件，在顺序结构的第 2 帧内添加一个条件结构，并将“大于”节点的输出端连接到条件结构选择端口。

(9) 切换到前面板窗口，选择“控件选板→布尔→圆形指示灯”控件，并将其命名为“素数”。切换到程序框图中，将对应于“圆形指示灯”控件的图标置于条件结构的外部。

(10) 在条件结构的真分支中添加一个布尔真常量，并与条件结构外的指示灯连接；同时，在条件结构的假分支中添加一个假布尔常量，并与条件结构外的指示灯连接，这样， $m$  为素数时，圆形指示灯亮；否则不亮。第 2 帧内真分支与假分支结构如图 6.17 和图 6.18 所示。

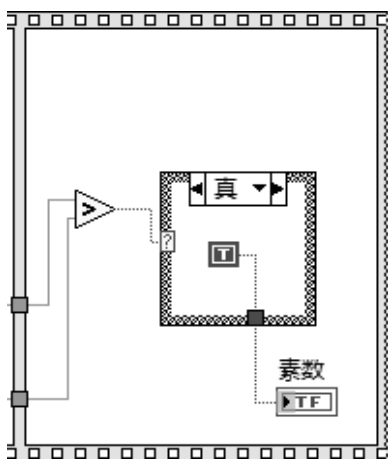


图 6.17 条件结构的真分支

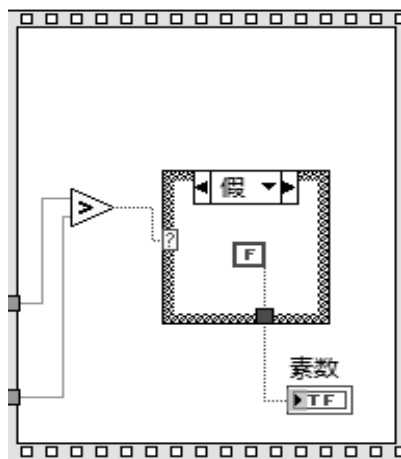


图 6.18 条件结构的假分支

(11) 切换到前面板，在数值  $m$  中任意输入一个整数，完整的程序框图和运行结果如图 6.19 所示。

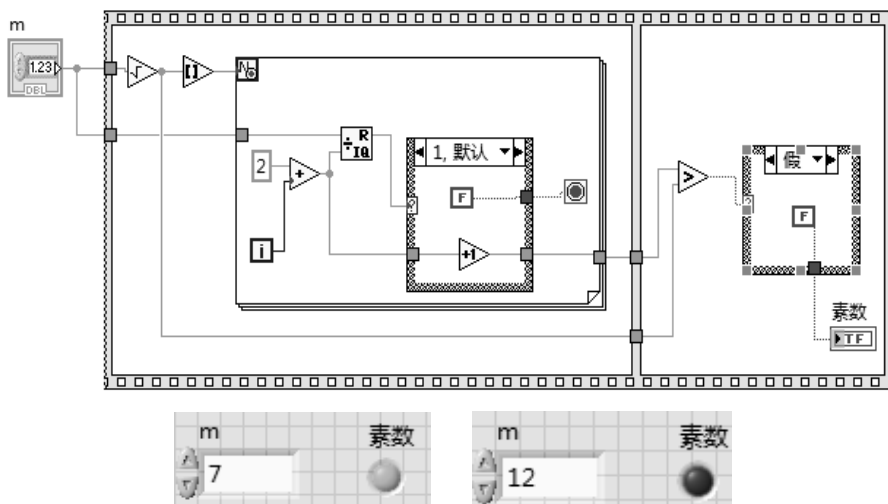


图 6.19 完整的程序框图和运行结果

**例 6-2** 使用三维曲面图显示  $z = \sin x \cos y + \sin(x - y) - \cos(x - y)$ ，其中  $x, y$  都在  $0 \sim 2\pi$  的范围内。

(1) 启动 LabVIEW2013，打开程序框图，选择函数选板中的“编程→结构→For 循环”控件，嵌套使用两个 For 循环结构，形成二维数组数据的输出。

(2) 选择“数学→初等与特殊函数→三角函数→正弦”、“数学→初等与特殊函数→三角函数→余弦”，分别将两个正弦函数和两个余弦函数置于内层 For 循环体中；选择“编程→数值→转换”转换为双精度浮点数值控件，将其置于内层 For 循环体中，将计数接线端  $i$  与其连接，使其与步长的数据类型相一致。

(3) 在“编程→数值”中选取相应的节点函数并连线。

(4) 切换到前面板窗口，选择控件选板中的“新式→图形→三维图形→三维参数图形”置于前面板中。切换到程序框图中，把  $x, y, z$  的坐标分别连接到三维曲面的  $x, y$  向量和  $z$  输入端口，完整的程序框图如图 6.20 所示。

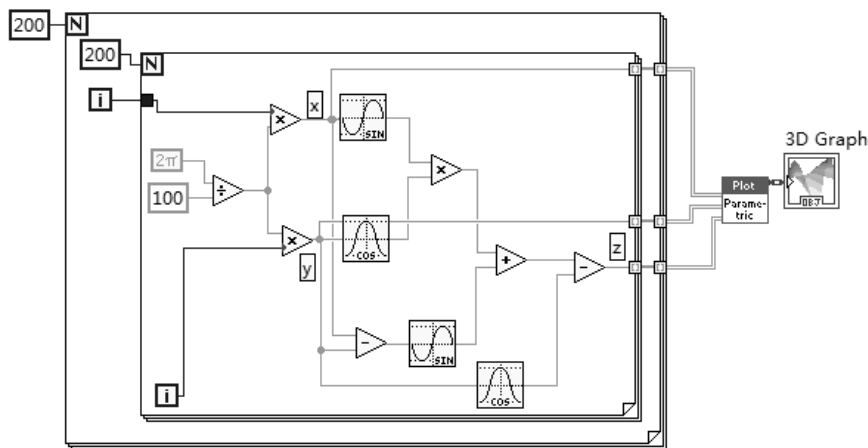


图 6.20 完整的程序框图

(5) 三维曲面  $z = \sin x \cos y + \sin(x - y) - \cos(x - y)$  如图 6.21 所示。按住鼠标左键可以通过旋转来改变三维曲面图的视点位置，如图 6.22 所示。

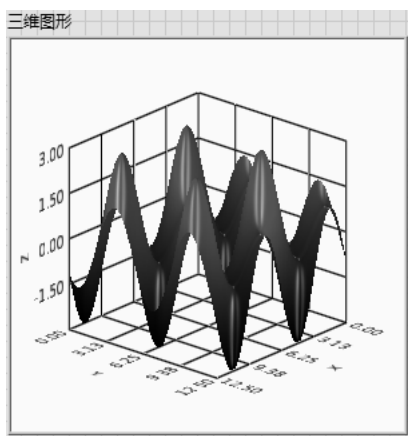


图 6.21 函数的三维曲面图

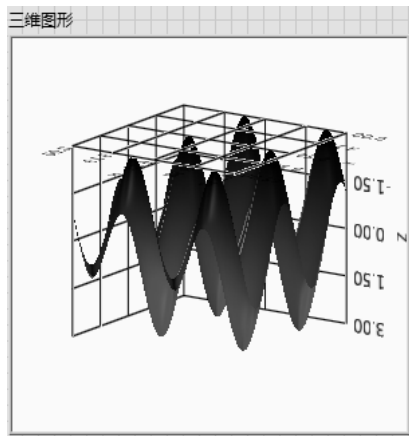


图 6.22 旋转后的三维曲面图

## 6.2.2 While 循环

### 1. 初识 While 循环

对于 For 循环，循环的次数是固定的，因为 LabVIEW 没有 Break 语句。在很多情况下用户需要让循环在满足某种条件时退出或继续运行，这时就需要借助于 While 循环。它是 LabVIEW 中最经常使用的一种程序结构，它在函数选板中的位置如图 6.23 所示。



图 6.23 While 循环在函数选板中的位置

当单击函数选板中对应的图标后，再在框图中按住鼠标左键向右下方拉动。估计大小合适的时候放开鼠标左键即可。

下面写一段最简单的代码画一个随机数曲线。首先在前面板放置一个波形图表和一个停止按钮（注意停止按钮的机械动作最好为释放时触发（Latch When Released），这样只有在鼠标单击按钮后才产生一个布尔值为 True 的脉冲，当 While 循环检测到这个脉冲时停止循环，同时按钮复位为 False）。然后在框图中放置一随机数发生器并与波形图表连接，再将按钮与 While 循环的条件接线端（Conditional Terminal）连接，如图 6.24 所示。

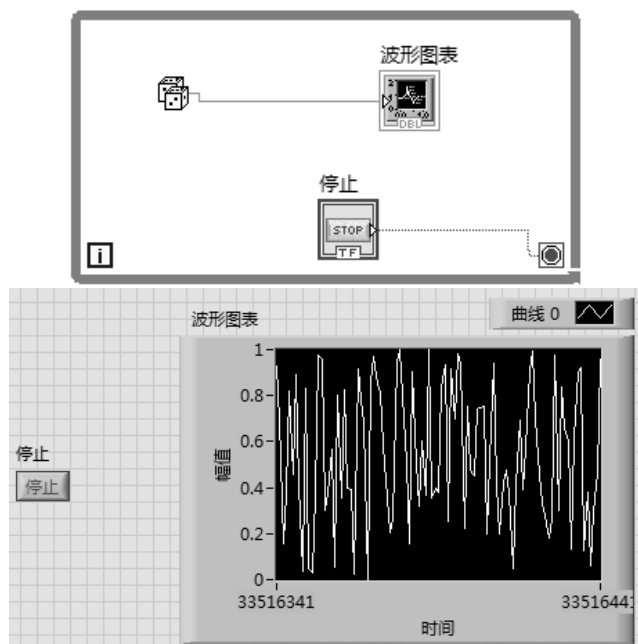




图 6.24 基于 While 循环的随机数曲线

其中条件接线端是循环终止条件的输入端子，默认为“真时停止 (Stop if true)”，即当输入布尔条件为 True 时终止循环，为 False 时继续运行。如果右击该端子选择“真时继续 (Continue if true)”，该端子变为，表示当输入布尔条件为 True 时继续运行，为 False 时终止循环。这两种情况是互为相反的循环。具体采取哪一种，则主要由程序的可读性决定。



While 循环体与外部代码交换数据的方式和 For 循环是一样的。但是如果以索引方式输入数组的话，While 循环的次数并不会由数组大小控制，它仍由布尔值控制。当循环次数超过数组大小时，将以数组最后一个元素作为输入。

## 2. 添加定时器

LabVIEW 在执行 While 循环时，如果用户没有设定循环时间间隔，那么它将以 CPU 的极限速度运行。按下 Ctrl+Alt+Delete 快捷键打开任务管理器，可以看到它几乎将 CPU 全部利用，如图 6.25 所示。正常情况这样做比较危险，因为这样可能会导致整个 LabVIEW 程序看上去跟“死掉”一样。



图 6.25 没有设定时间间隔情况下的 CPU 利用率

在很多情况下没有必要让 While 循环以最大的速度运行，所以最好给 While 循环加上时间间隔。有两种方法：一种是在每个循环中添加一个等待时间，只有在等待完后才运行下一个循环；另一种方法是使用定时循环（Timed Loop）。有两种等待定时器，它们都在函数选板的“编程→定时（Programming|Timing）”选项下。一种是“等待（Wait(ms)）”，即等待一个指定的时间；另一种是“等待下一个整数倍毫秒（Wait Until Next ms Multiple）”，即一直等到定时器时间是设定时间的整数倍为止。在绝大多数情况下，这两种定时器是一样的。下面将它添加到程序的 While 循环中去，如图 6.26 所示。从图中可以看到 CPU 利用率已经大幅下降到正常情况了。因此，在写 While 程序时千万别忘了加上一个定时器。

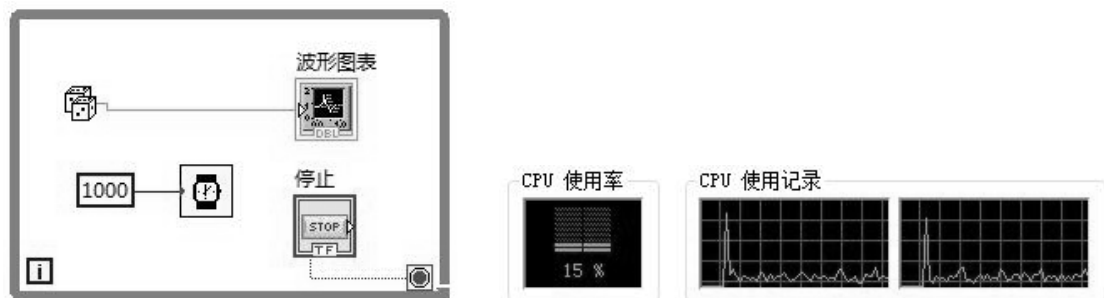


图 6.26 设定时间间隔后的 CPU 利用率

### 3. 使用移位寄存器和反馈节点

和 For 循环一样，在 While 循环中也会经常遇到下一个循环使用上一个循环产生的结果的情况。同 For 循环一样，也可以使用移位寄存器和反馈节点来实现。实现方法也是一样的，这里就不再重复了。

**例 6-3** 计算 e 的近似值： $e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!}$ （精确到  $\frac{1}{n!} < 10^{-6}$  为止）。

（1）启动 LabVIEW 2013，打开前面板，选择控件选板中的“新式→数值→数值显示控件”，将两个“数值显示控件”置于前面板中，并分别命名为“e 近似值”和“临界界次 N”。

（2）切换到程序框图窗口，选择函数选板“编程→结构→For 循环”控件，将其置于程序框图中，并在 For 循环右边框上右击创建位移寄存器，在 For 循环框内，通“新式→数值”添加“加 1”控件和“乘”控件；在 For 循环左侧位移寄存器上右击创建“常数”，给该常数赋值为“1”。

（3）在 For 循环框图外通过“编程→结构→While 循环”放置一个 While 循环结构体，使得两个循环结构形成嵌套循环，While 循环的循环次数加 1 作为“临界界次 N”的显示，在 While 循环通过“新式→数值→倒数”和“新式→比较→小于”放置“倒数”和“小于”控件，并在“小于”控件前创建常数“1E-6”作为精确度的限定值，各控件连接如图 6.27 所示。

（4）选择“编程→结构→For 循环”控件，放置 While 循环体后面利用自动索引功能实现各阶乘倒数和的计算，并将“e 近似值”与其相连，运行程序。执行结构和完整的程序框图如图 6.28 所示。

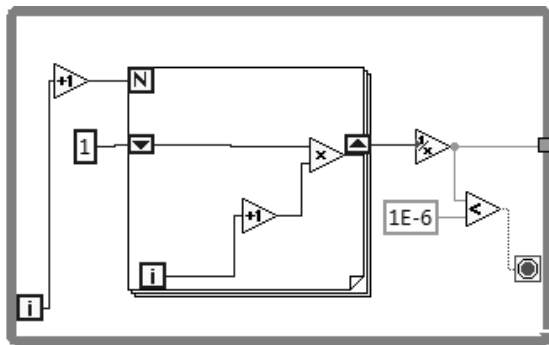


图 6.27 嵌套循环体结构

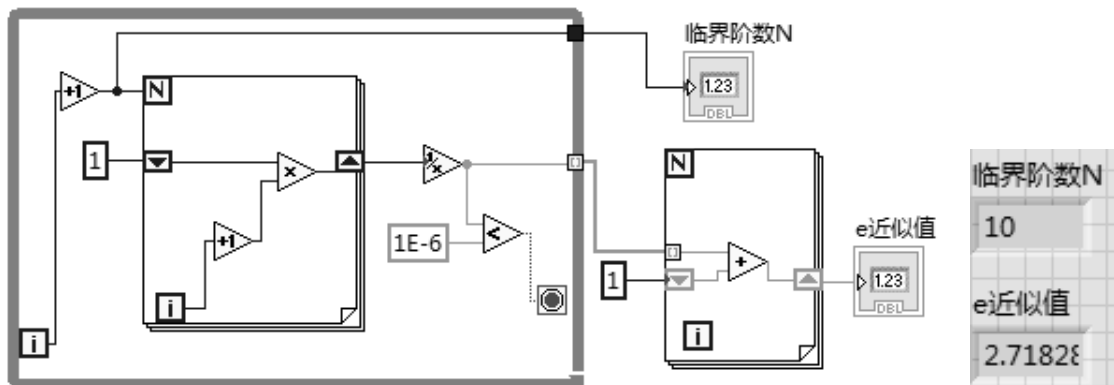


图 6.28 执行结构和完整的程序框图

## 6.3 层次结构

### 6.3.1 条件结构

Case 在英文里的意思是“在某种特定情况下”，因此条件（Case）结构的意思就是在某种特定情况下执行某段特定程序。它等价于 C 语言里的 if...else...语句或 Switch 语句。它在函数选板中的位置如图 6.29 所示。当鼠标单击函数选板中对应的图标后，再在框图中按住鼠标左键向右下方拉动，估计大小合适的时候放开鼠标左键即可。

#### 1. 等价于 if...else...语句的 Case 结构

当某种条件得到满足时或得不到满足时执行某段特定的程序，也就是当某个布尔值为真或假时执行某段特定的程序就是 if...else...语句的功能，因此当 Case 结构的输入为布尔值时就对应 C 程序中的 if...else...语句。

Case 结构的输入端子默认就是布尔输入，因此不需要任何改动，只需要把当作条件的布尔值与输入端子连接即可，如图 6.30 所示，该程序等效的伪代码如下。



图 6.29 条件结构在函数选板中的位置

```

If (a%2==0)
b=a+2;
Else
b=a+1;
End
    
```

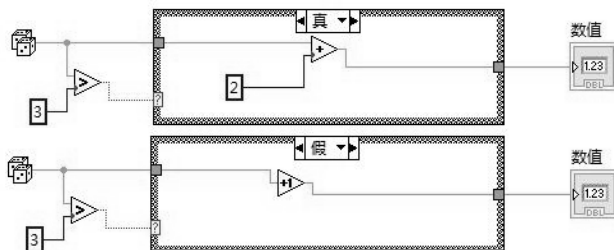



图 6.30 等价于 if...else...语句的条件结构



通过单击当前条件框中的小黑三角来切换 True 和 False。当显示为 True 时表明当前 Case 框中程序只有在输入为 True 时才执行，当显示为 False 表明当前框中程序只有在输入为 False 时才执行。

## 2. 等价于 Switch 语句的 Case 结构

当 Case 结构的输入端子不是布尔变量的时候，就等价于 C 语言中的 Switch 语句了。在编写程序的过程中需要添加 Case 分支，可以通过右击条件框，选择“在后面添加分支（Add Case After）”或“在前面添加分支（Add Case Before）”选项来实现，同理，如果要删除的话选择“删除本分支（Delete This Case）”选项即可。至于该右击菜单中的其他功能，读者不妨自己去试试看。

**例 6-4** 创建一个温度报警系统，连续以每 100ms 一次的速率测量温度，如果温度高于或低于设定的温度范围，警告灯点亮，同时蜂鸣器报警，工作状态栏显示“超量限”；若检测温度在量程范围内，正常指示灯亮，同时工作状态栏显示“正常”信息。

(1) 启动 LabVIEW2013，打开前面板，选择控件选板中的“新式→数值→数值输入控件”，将两个“数值输入控件”置于前面板中，分别命名为“量程上限”、“量程下限”；选择“新式→布尔→圆形指示灯”、“新式→布尔→方形指示灯”、“新式→字符串与路径→字符串显示控件”、“新式→数值→温度计”和选择“新式→布尔→停止按钮”置于前面板中，分别命名为“正常”、“警告”、“工作状态”、“温度计”和“停止”；选择“新式→波形→波形图”控件置于前面板中，并将图例中的波形设置为三条不同颜色的曲线。

(2) 切换到程序框图，选择函数选板中的“编程→比较→判定范围并强制转换”、“编程→布尔→非”控件，置于程序框图中；选择“编程→数组→创建数组”函数，将量程上限、温度计和量程下限三个控件整合成一个数组，将“创建数组”函数的输出与波形图相连，如图 6.31 所示。

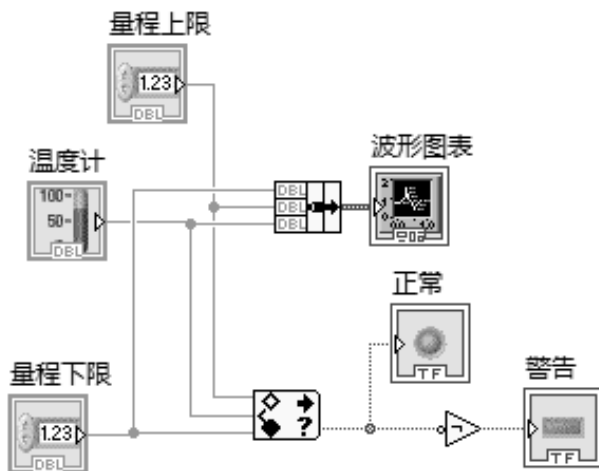


图 6.31 温度范围设定

(3) 选择“编程→结构→条件结构”控件置于程序框图中，将判定控件的“范围内？”输出与条件结构的“选择器输入端”相连，通过判定温度的高低来决定条件结构的执行程

序。在条件结构“真”分支中，添加一个“字符串常量”控件，命名为“正常”，在条件结构“假”分支中，添加一个“字符串常量”控件，命名为“超量限”，这两个字符串控件都通过条件结构的边框与前面板中对应的“工作状态”控件相连，同时选择“编程→图形与声音→蜂鸣器”，置于“假”条件结构分支中，如图 6.32 所示。

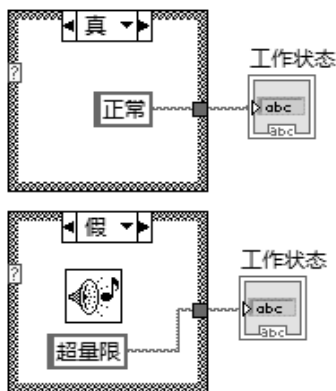
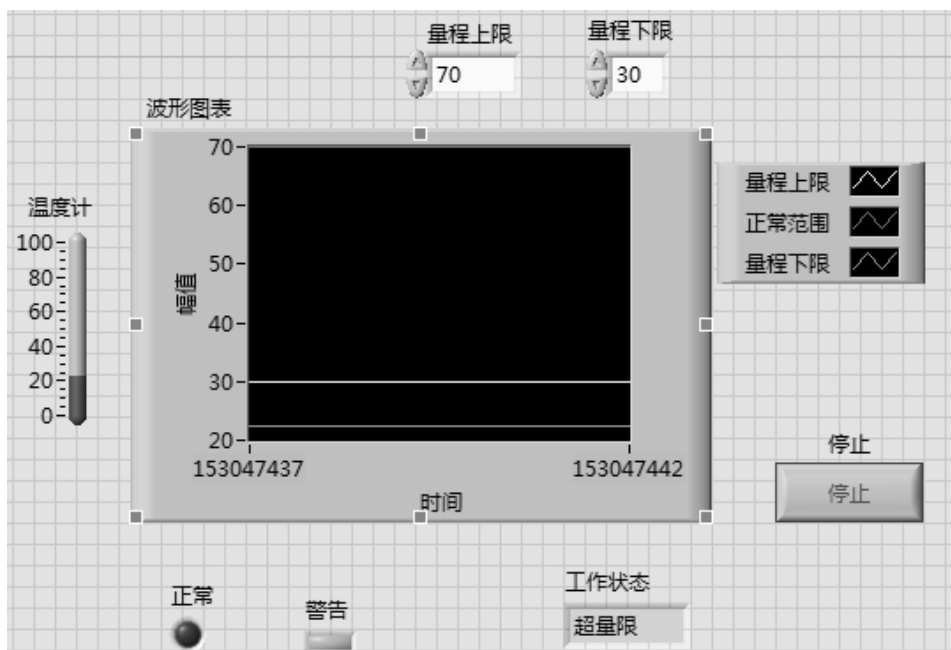


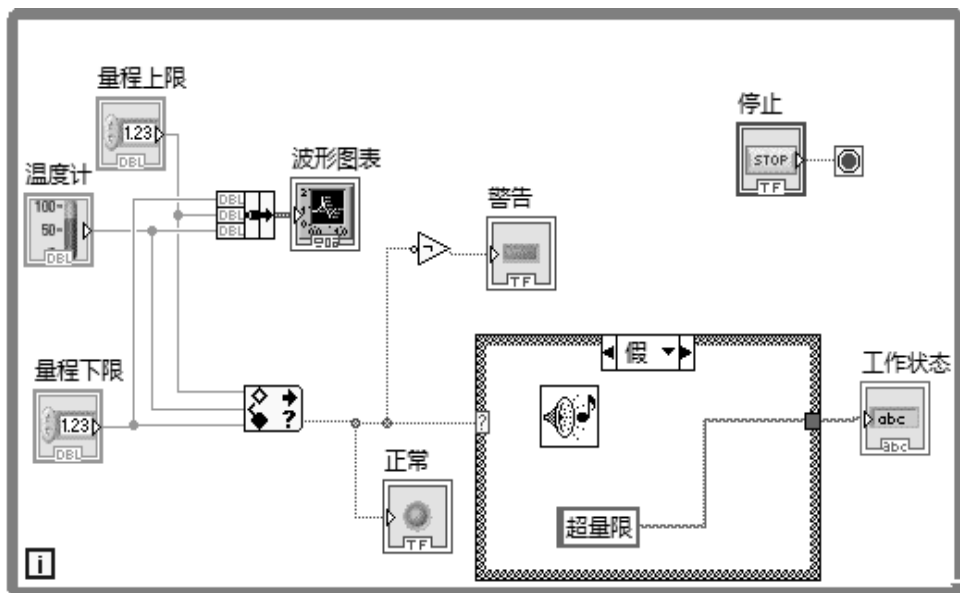
图 6.32 条件结构分支

(4) 选择“编程→定时→等待下一个整数倍毫秒”，放置在程序框图中，并通过鼠标右键创建“常量”，设置为 100。选择“编程→结构→While 循环”，置于程序框图中，将程序框图中所有的控件放置在 While 循环中，将“停止”按钮的输出端与其判断条件端相连。通改变“温度计”的值来观察三条曲线的波形，前面板和程序框图如图 6.33 所示。



(a) 前面板

图 6.33 温度报警系统



(b) 程序框图

图 6.33 温度报警系统 (续)

### 6.3.2 事件结构

#### 1. 事件结构 (Event Structure) 的常用方法

考虑用现有的知识编写一个简单的单击计数器，即当用户单击一个按钮时，计数器加 1。

1. 目前为止，唯一的办法就是通过 While 循环和 Case 结构不断查询这个按钮是否被单击，如果被单击的话，计数器加 1，否则什么都不干，如图 6.34 所示。

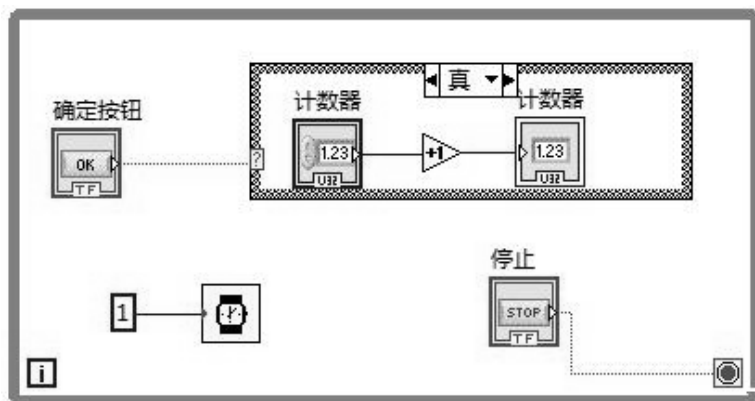


图 6.34 基于 While 循环和 Case 结构的单击计数器

分析该程序可以看出，这个程序在没有用户单击的情况下完全都是在“空转”，浪费了大量的 CPU 资源，而且当“事件”发生太快时可能会被忽略。因此 LabVIEW 提供了事件结构来解决这个问题，即仅当“事件”发生时，程序才作相应的响应，类似于 Visual C++ 或

VB 编程中的事件。

通过事件结构，程序可以变得很简单，并降低了 CPU 的利用率。当多个事件发生时会造成事件队列，直到每个事件对应的代码都被执行为止，因此不会有事件被漏掉的情况，事件结构在函数选板中的位置如图 6.35 所示。



图 6.35 事件结构

同 Case 结构类似，当鼠标单击函数选板中对应的图标后，再在框图中按住鼠标左键向右下方拉动，估计大小合适的时候放开鼠标左键即可，方框中的程序就是对应事件发生时被执行的程序。

默认的时间为超时（Timeout）事件。它表示当指定的超时时间内没有任何该事件结构所定义的其他事件发生时，触发超时事件。

Events 栏中的某些事件，如 Mouse Down 左侧有两种箭头：一种是绿色箭头，一种是红色箭头。其中绿色的是 Notify 事件，即当事件发生时通知程序运行该事件所对应的代码，大部分情况下都使用这种事件。红色箭头表示该事件是 Filter 事件，即当事件发生时可以选择是不是过滤该事件。

如果把上面的程序改写为两个按钮，即当单击其中任何一个按钮时，计数器都加 1。但我们没有必要再复制这样一个事件，只需要把 hit me 所对应的时间再添加一个触发源即可。编辑该事件，在出现的事件编辑器中单击“+ 添加事件”来添加该事件代码的触发源，即当其中任何一个事件激发时都会执行该事件框中的代码。

## 2. Filter 事件

事件编辑器中事件左侧为红色箭头的是 Filter 事件。顾名思义，当该事件发生时，用户可以过滤甚至完全放弃该事件将触发的活动。例如，通过 Filter 事件，当用户试图关闭一个 VI 时，将弹出一个对话框，提示该 VI 不能被关闭，并放弃关闭该 VI，这在很多情况下是非常有用的。

**例 6-5** 利用事件结构编写密码登录程序（当用户输入的密码是“123456”时，弹出“密码正确登录成功”对话框，单击“确定”按钮程序停止；如果密码错误，则弹出“密码错误请重新输入”对话框，单击“确定”按钮后继续运行）。

（1）启动 LabVIEW 2013，打开前面板，选择控件选板中的“新式→字符串与路径→字符串输入”控件，置于前面板中，命名为“请输入密码”，鼠标右键单击控件，从弹出的快捷菜单中选择“密码显示”命令，这样输入的字符密码显示的是“\*”；选择“新式→布尔→确定按钮”置于前面板，命名为“登录”，将其作为“登录”事件的触发源。

（2）切换到程序框图，选择函数选板中的“编程→结构→事件结构”控件，放置在程序框图中，在事件结构体上右击，从弹出的快捷菜单中选择“编辑本分支所示处理的事件”命令，弹出“编辑事件”对话框，修改触发源“登录”事件为“值改变”，如图 6.36 所示。

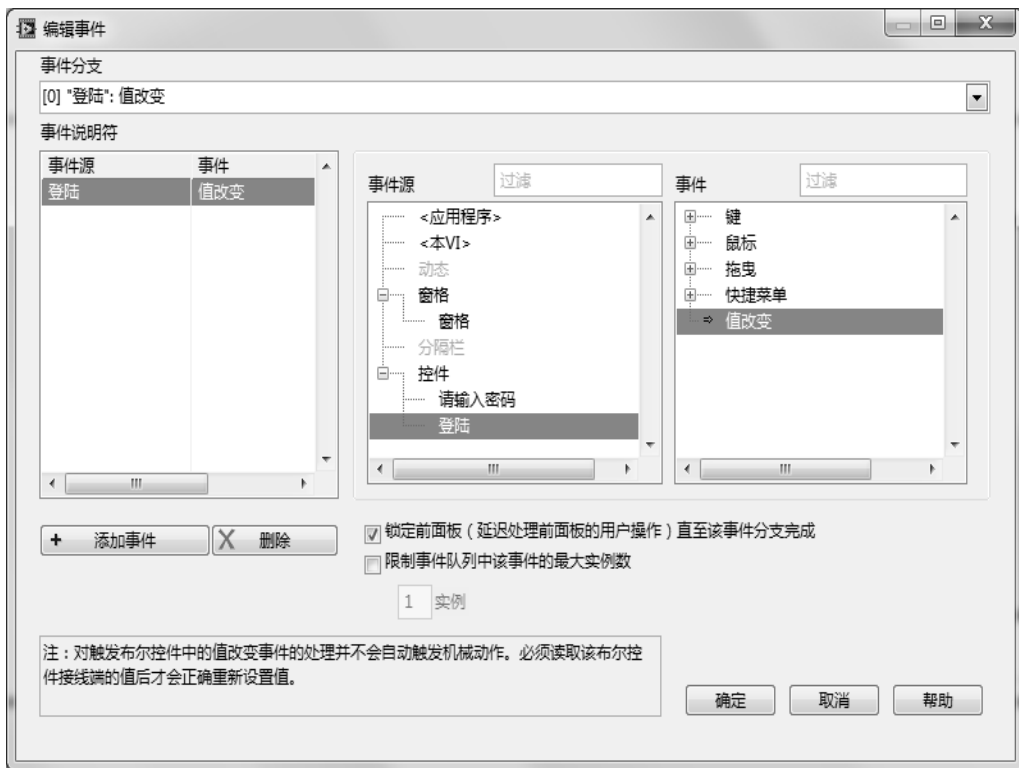


图 6.36 “编辑事件”对话框

(3) 将程序框图中的“请输入密码”、“登录”控件放置在登录事件分支中，选择“编程→比较→等于”控件，放置在登录事件分支中，一端连接“请输入密码”的输出端，另一端通过右键创建一个字符串常量“123456”，作为登录的设定密码。

(4) 在事件结构体外创建一个条件结构，选择“编程→结构→条件结构”控件，放置在程序框图中，条件端口与“等于”控件的输出口连接；选择“编程→对话框与用户界面→单按钮对话框”控件，放置在条件结构“真”分支中，设置显示文本为“密码正确登录成功”，同样在条件结构“假”分支中放置一个“单按钮对话框”控件，设置显示文本为“密码错误请重新输入”，即当输入的密码等于“123456”时，输入为“真”，反之，则输出为“假”。条件结构如图 6.37 所示。

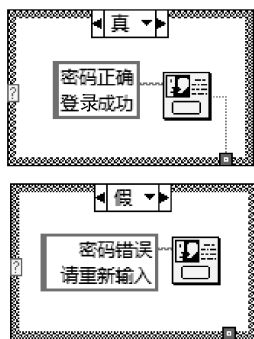


图 6.37 条件结构

(5) 选择“编程→结构→While 循环”，置于程序框图中，将程序框图中所有的控件放置在 While 循环中，循环条件输入端口与“真”条件分支的对话框输入端口连接，在“假”分支的数据通道上右击，从弹出的快捷菜单中选择“未连接时使用默认”命令。完整的程序框图如图 6.38 所示。

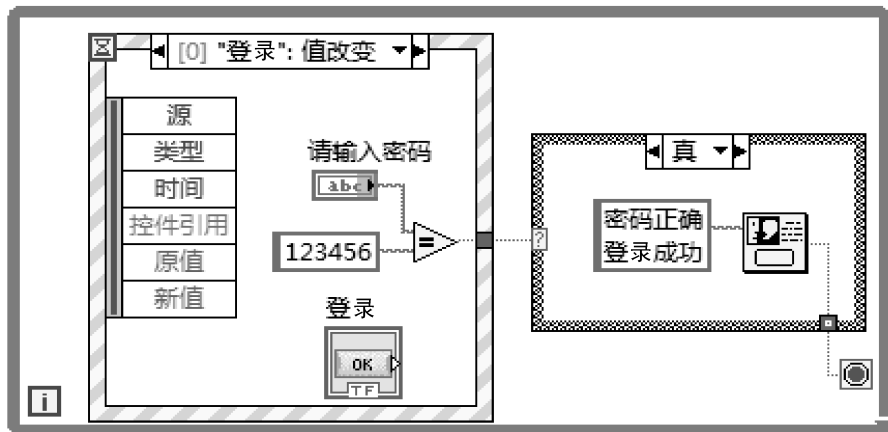


图 6.38 完整的程序框图

(6) 运行程序，正确输入密码时弹出的对话框和错误输入密码时弹出的对话框如图 6.39 所示。



图 6.39 密码登录弹出的对话框

### 6.3.3 使能结构

使能结构用来控制程序是否被执行。有两种使能结构，最常用的一种是框图使能结构 (Diagram Disable Structure)，其功能类似于 C 语言中的 `/*...*/`，用于大段地注释程序。另一种是条件使能结构 (Conditional Disable Structure)，用于通过外部环境变量来控制代码是否执行，类似于在 C 语言中通过宏定义来实现条件编译。这两种使能结构在函数选板中的位置如图 6.40 所示，其使用方法与 Case 结构类似。



图 6.40 两种使能结构在函数选板中的位置

### 1) 框图使能结构——注释程序框图

在 C 语言中, 如果不想让一段程序执行, 可以通过 `/*...*/` 的方法注释掉。在虚拟仪器中有框图使能结构来实现真正的注释功能, 只要将需要注释的代码段放在使能结构中, 并使之成为“禁用 (Disable)”状态。如果需要重新使能这段代码, 只需要右击使能结构边框, 选择“启用本子程序框图 (Enable This Subdiagram)”使能该段即可。注意, 如果按同样方法再把它 Disable 时, 一定要将另一个 Case 条件框使能, 否则程序不能被执行。

### 2) 条件使能结构

在 C 语言中, 程序员可以通过宏定义的方法通过外部条件控制某段程序是否执行。在 LabVIEW 中的条件使能结构也提供了类似的功能。通过定义外部环境变量为真或假来控制代码是否执行。此外, 还可以通过判断当前操作系统的类型来选择执行哪段代码。

环境变量只有在项目 (Project) 中才能使用。通过定义整个项目的环境变量, 该项目下所有的 VI 都可以被这些环境变量控制。如果该项目下的 VI 脱离项目单独运行的话, 将不受环境变量的控制。

条件编辑对话框中还可以选择 OS、CPU、Target Type。它可以用来判断当前的操作系统、CPU 类型等。可以通过右击“我的电脑 (My Computer)”并选择“属性 (Properties)”选项, 在出现的对话框中选择“条件禁用符号 (Conditional Disable Symbols)”来对这些系统环境变量进行编辑。这在某些情况下也是很有用的, 比如可以根据当前的 CPU 的类型选择不同的算法。

## 6.4 公式节点

公式节点 (Formula Node) 是一种程序结构, 通过公式节点, 用户不仅可以实现复杂的数学公式, 还能通过文本编程写一些基本的逻辑语句, 如 `if...else...`、`case`、`while` 循环之类的语句。可以这么说, 公式节点基本上弥补了图形化开发语言相对于文本语言的缺陷。公式节点的语法与 C 语言基本相同。赋值语句必须以分号结束, 注释的内容通过 `/*...*/` 封闭起来。

### 6.4.1 复杂公式的实现

在 LabVIEW 中, 如果遇到  $y = \frac{a^3 + \sqrt{a} + b^5 - \cos b}{e^a - \operatorname{tg} b + \sin a}$  这样结构复杂的公式, 就需要公式节点来帮忙了。公式节点在函数选板中的位置如图 6.41 所示。





图 6.41 公式节点在函数选板中的位置

当鼠标单击函数选板中对应的图标后，再在框图中按住鼠标左键向右下方拉动，估计大小合适的时候放开鼠标左键即可。在框图上出现的公式节点左边框上右击，选择“添加输入 (Add Input)”选项，接着输入  $a$ ，再重复一次输入  $b$ 。再右击右边框，选择“添加输出 (Add Output)”选项，输入  $y$ 。然后在这些输入输出端子上右击，选择“创建→输入控件 (Create|Control)”或“创建→显示控件 (Create|Indicator)”选项来创建输入/输出控件。下面就可以直接将公式文本写在公式节点框中了。公式文本为 “ $y=(a**3+sqrt(a)+b**5-cos(b))/(exp(a)-tan(b)+sin(a))$ ；”，如图 6.42 所示。

公式节点不仅能实现一个公式，还能同时实现多个公式，只需要增加输入/输出端子就可以了，如图 6.43 所示。

公式节点可以使用的数学函数和操作符与表达式节点完全相同。数学函数有：abs、acos、acosh、asin、asinh、atan、atan2、ceil、cos、cosh、cot、csc、exp、expm1、floor、getexp、

getman、int、intrz、ln、lnp1、log、log2、max、min、mod、pow、rand、rem、sec、sign、sin、sinc、sinh、sizeofDim、sqrt、tan、tanh。支持的运算符有：=、+=、-=、\*=、/=、>>=、<<=、&=、^=、|=、%=、\*\*=、+、-、\*、/、^、!、==、<、>、<=、>=、&&、||、&、!、%、\*\*、!、++、--、~。

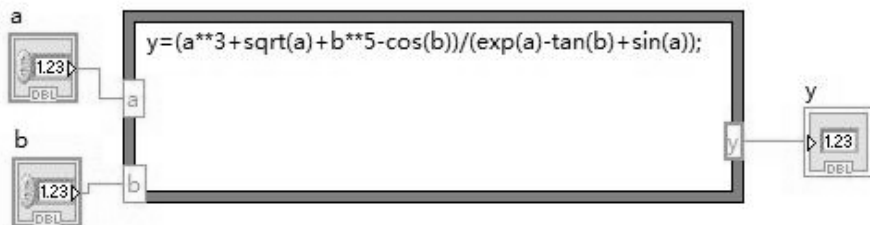


图 6.42 公式节点

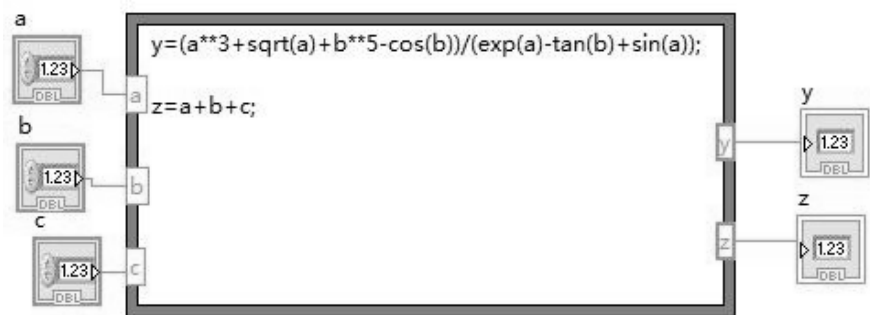


图 6.43 公式节点中可以包含任意数量的公式

## 6.4.2 文本编辑语言的实现

公式节点文本编辑语言的语法与 C 语言非常接近，但是只能实现基本的逻辑流程和运算，不能对文件或设备进行操作或通信，没有输入/输出语句。主要有以下几种语句：变量声明型赋值语句、条件语句、循环语句、Switch 语句、控制语句。注释方法为//或/\*...\*/，下面对它们逐个进行介绍。

### 1) 变量声明

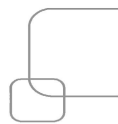
公式节点支持的数据类型有：float、float32、float64、int、int8、int16、int32、uInt8、uInt16、uInt32。

float a; //声明浮点型数据

uInt32 y[10]; //声明数组

### 2) 赋值语句

赋值符号有：=、+=、-=、\*=、/=、>>=、<<=、&=、^=、|=、%=、\*\*=。具体含义和 C 语言一致。例如，“a=b;”。



```
a&=b; //等价于 a=a&b;
```

### 3) 条件语句

if 语句举例如下:

```
if(a>0)
b=a;
```

if...else 语句举例如下:

```
If(a>0)
{b=a;
c=2*a;
}
Else
b=2*a;
```

### 4) 循环语句

do...While 语句举例如下:

```
do
{a++;
b--;
}while(b>=0);
```

While 语句举例如下:

```
While(b>=0)
{a++;
b--;
}
```

For 语句举例如下:

```
For(i=0;i<10;i++)
{y[i]=i;
b+=i;
}
```

Break、Continue 语句用于当某种条件满足时终止循环或让循环立即重新从头开始。

Break 语句还能用在 Switch 语句中, 含义仍然一样。

举例如下:

```
for(i=0;i<10;i++)
{y[i]=i;
```

```

if(y[i]>5) Break; //当 y[i]>5 的时候，终止循环
b+=i;
}

```

### 5) Switch 语句

Switch 语句举例如下：

```

Switch(a)
{Case 0:b=a+1;break;
Case 1:b=a+2;break;
Case 11 b=a+7;break;
Default: b=0;
}

```

在实现复杂算法方面，文本编辑语言要更简洁、更容易管理代码、更容易读懂。而图形编程在实现复杂算法时就显得杂乱、代码冗余、不易读懂。因此，无论是图形编程还是文本编程都有它自己的优势。图形编程的优势在于其数据流的形式让工程师能直接从大局上考虑整个程序，思路可以很清晰。但是当遇到逻辑复杂的算法或公式时，建议还是使用公式节点。因此可以这么说，公式节点基本上弥补了图形化开发语言相对于文本语言的缺陷。

## 6.5 结构函数传递函数

在两次循环之间传输数据时使用反馈节点和移位寄存器。其中，反馈节点和只有一个左端子的移位寄存器的功能相同。

### 6.5.1 反馈节点

反馈节点主要保存 VI 或循环上一次的运行数据，反馈节点使用连线至初始化接线端的值作为第一次程序框图执行或循环的初始值。如初始化接线端未连线任何值，该 VI 使用数据类型的默认值。反馈节点可保存上一次执行或循环的结果，可通过启用接线端来启用或禁用反馈循环。如设置启用接线端为 TRUE，反馈节点按用户在属性对话框或节点快捷菜单中的配置运行。如启用接线端设置为 FALSE，反馈节点将忽略输入值并输出接线端为 TRUE 时最近一次循环的值。反馈节点将一直输出该值，直到反馈节点的启用接线端转换为 TRUE 为止。在默认状态下，反馈节点仅保存上一次执行或循环所得的数据。通过使节点延迟多次执行或循环输出，可配置反馈节点存储  $n$  个数据采样。如增加延迟值，使其大于一次执行或循环的执行时间，在延迟结束前，反馈节点仅输出初始化接线端的值。然后，反馈节点可按

顺序输出存储值。反馈节点边框上的数字为延迟。

循环中一旦连线构成反馈，就会自动出现反馈节点箭头和初始化端子。使用反馈节点需要注意其在选项板上的位置，若在分支连接到数据输入上，则反馈节点把每个值都传递给数据输入端；若在分支连接到数据输入端的连线之后把反馈节点放到连线上，反馈节点把每个值都传回 VI 或函数的输入，并把最新的值传递给数据输入端。

如图 6.44 所示，求  $n!$  的值，由于本例需要访问以前循环的数据，所以要使用移位寄存器或反馈节点。图 6.44 所示是使用移位寄存器来实现计算  $n!$  的功能。

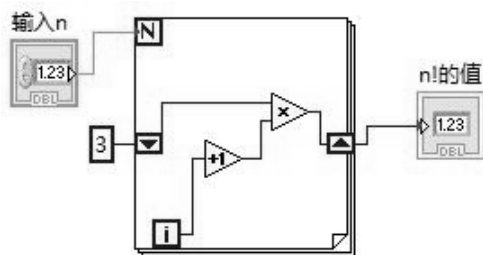


图 6.44 使用带移位寄存器的 For 循环求  $n!$

因为反馈节点和只有一个左端子的移位寄存器的功能相同，所以可使用反馈节点来完成程序，具体程序框图如图 6.45 所示。

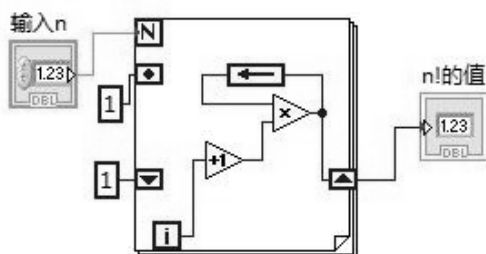


图 6.45 使用带反馈节点的 For 循环求  $n!$

本例如果使用 While 循环来实现，则需要构件条件来判定其什么时候执行循环，此时可以通过自增的数是否小于输入数来判断是否继续执行，如图 6.46 所示。

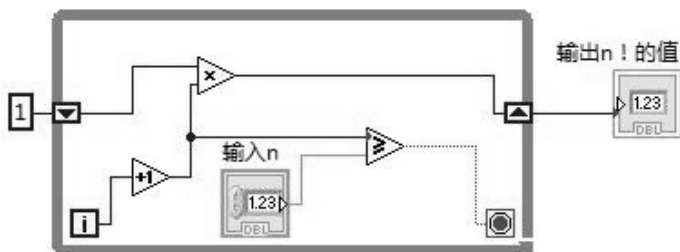
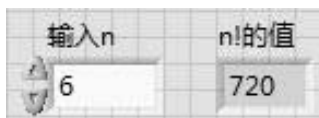


图 6.46 使用带移位寄存器的 While 循环求  $n!$

对于上面 3 个程序框图，当输入 6 时，输出结果均为 720，如图 6.47 所示。

图 6.47  $n!$  的输出结果

## 6.5.2 移位寄存器

移位寄存器是 LabVIEW 循环结构中的一个附加对象，也是一个非常重要的方面，其功能是把当前循环完成的某个数据传递给下一个循环开始。移位寄存器的添加可以通过在循环结构的左边框或右边框上弹出的快捷键获得，在其中选择添加移位寄存器，如图 6.48 所示。



图 6.48 在 For 循环中添加移位寄存器

图 6.49 所示是添加移位寄存器后的程序框图。



图 6.49 添加了移位寄存器的程序框图

右端子在每次完成一次循环后存储数据，移位寄存器将上次循环的存储数据在下次循环开始时移动到左端子上，移位寄存器可以存储任何数据类型，但连接在同一个寄存器端子上的数据必须是同一种类型，移位寄存器的类型与第一个连接到其端子之一的对象数据类型相同。

如计算  $1+2+3+4+5$  的值，由于是累加的结果，所以用到了移位寄存器。需要注意的是，由于 For 循环是从 0 执行到  $N-1$ ，所以输入端赋予了 6，移位寄存器赋予了初值 0。具体程序框图和前面板显示如图 6.50 所示。

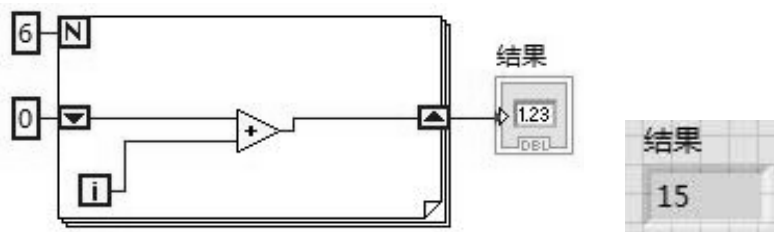


图 6.50 计算  $1+2+3+4+5$  的值

若上例中没有添加移位寄存器则只输出 5，因为此时没有累加结果的功能，如图 6.51 所示。

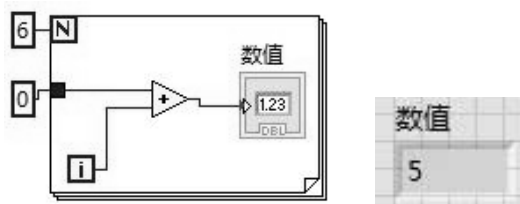


图 6.51 不添加移位寄存器的结果

又如求  $0\sim 99$  之间偶数的总和。由于 For 循环中的默认递增步长为 1，此时根据题目要求步长应变为 2，具体程序框图和前面板如图 6.52 所示。

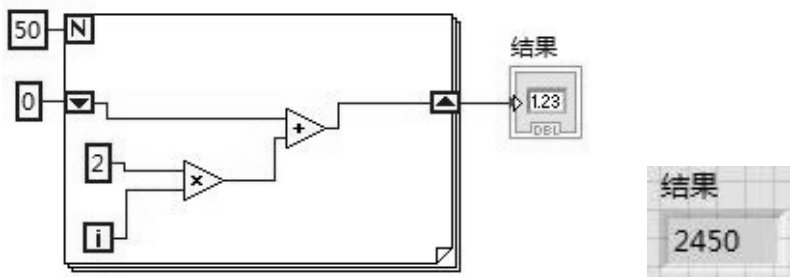


图 6.52 计算  $0\sim 99$  中偶数的和

在使用移位寄存器时应注意初始值问题，如果不给移位寄存器指定明确的初始值，则左端子将在对其所在循环调用之间保留数据，当多次调用包含循环结构的子 VI 时会出现这种情况，需要特别注意。如果对此情况不加考虑，可能引起错误的程序逻辑。

在一般情况下应为左端子明确提供初始值，以免出错，但在某些场合，利用这一特性也

可以实现比较特殊的程序功能。除非显式的初始化移位寄存器，否则当第一次执行程序时移位寄存器将初始化为移位寄存器相应数据类型的默认值，若移位寄存器数据类型是布尔型，初始化值将为假；若移位寄存器数据类型是数字型，初始化值将为零，但当第二次开始运行时，第一次运行时的值将为第二次运行时的初始值，依此类推。例如当不给图 6.52 中的移位寄存器赋予初值时即如图 6.53 所示，当第一次执行时，输出为 2450，再运行时输出为 4900，这就是因为左端子在循环调用之间保留了数据。

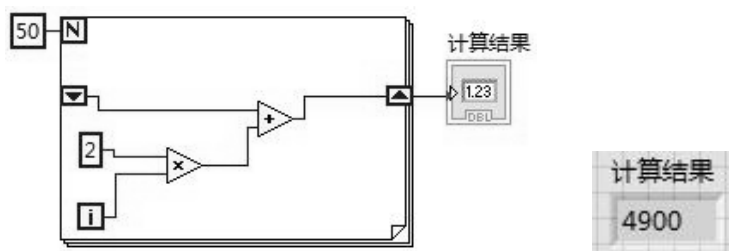


图 6.53 计算 0~99 中偶数的和，移位寄存器不赋予初值的情况

移位寄存器也可以添加多个移位寄存器，可通过多个移位寄存器保存多个数据，如图 6.54 所示，该程序框图用于计算等差数列  $2n+2$  中  $n$  取 0, 1, 2... 时的乘积。

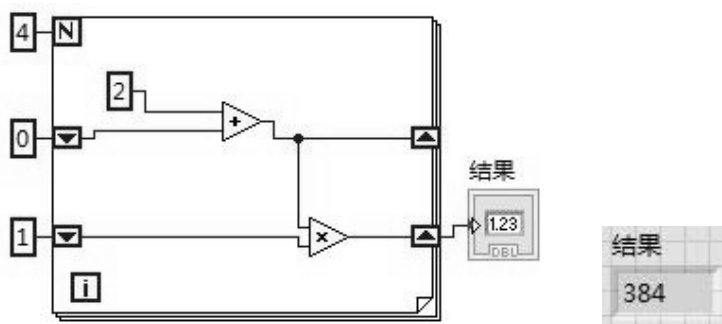


图 6.54 计算等差数列的乘积

在编写程序时有时需要访问以前多次循环的数据，而层叠移位寄存器可以保存以前多次循环的值，并将值传递到下一次循环中。创建层叠移位寄存器，可以通过使用右键单击左侧的接线端并从其中选择添加元素来实现，如图 6.55 所示。层叠移位寄存器只能位于循环左侧，因为右侧的接线端仅用于把当前循环的数据传递给下一次循环。

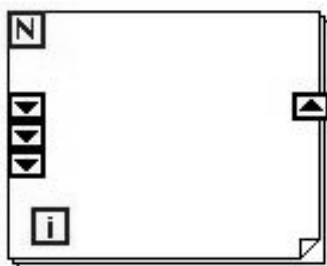


图 6.55 层叠移位寄存器



## 6.6 定时循环

定时循环和定时顺序结构都用于在程序框图上重复执行代码块或在限时及延时条件下按特定顺序执行代码，定时循环和定时顺序结构都位于定时结构子选板中，如图 6.56 所示。

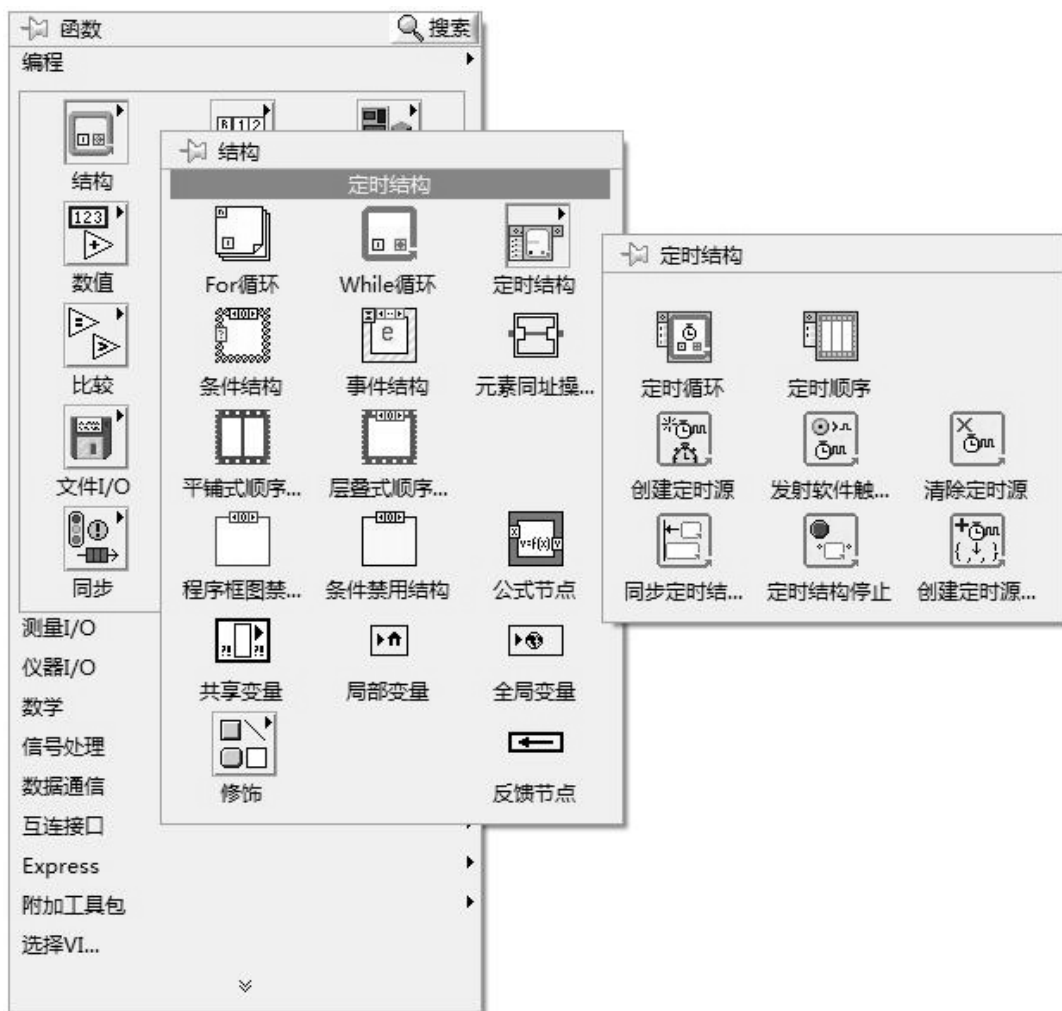


图 6.56 定时结构子选板

### 6.6.1 定时循环和定时顺序结构

添加定时循环与添加普通的循环一样，通过定时循环可以设定精确的定时代码，协调多个对时间要求严格的测量任务，并定义不同优先级的循环，以创建多采样的应用程序。与 While 循环不同，定时循环不要求与“停止”接线端相连。如不把任何条件连接到“停止”接线端，循环将无限运行下去。定时循环的执行优先级介于实时优先级和高优先级之间。这

意味着在一个程序框图的数据流中，定时循环总是在优先级不是实时的 VI 前执行。若程序框图中同时存在优先级设为实时的 VI 和定时顺序，将导致无法预计的定时行为出现。

对于定时循环，双击输入端子，或右键单击输入节点并从快捷菜单中选择配置输入节点，可打开“配置定时循环”对话框。在对话框中可以配置定时循环的参数，也可以直接将各参数值连接至输入节点的输入端进行定时循环的初始配置，如图 6.57 所示。图 6.58 所示为定时循环的结构。

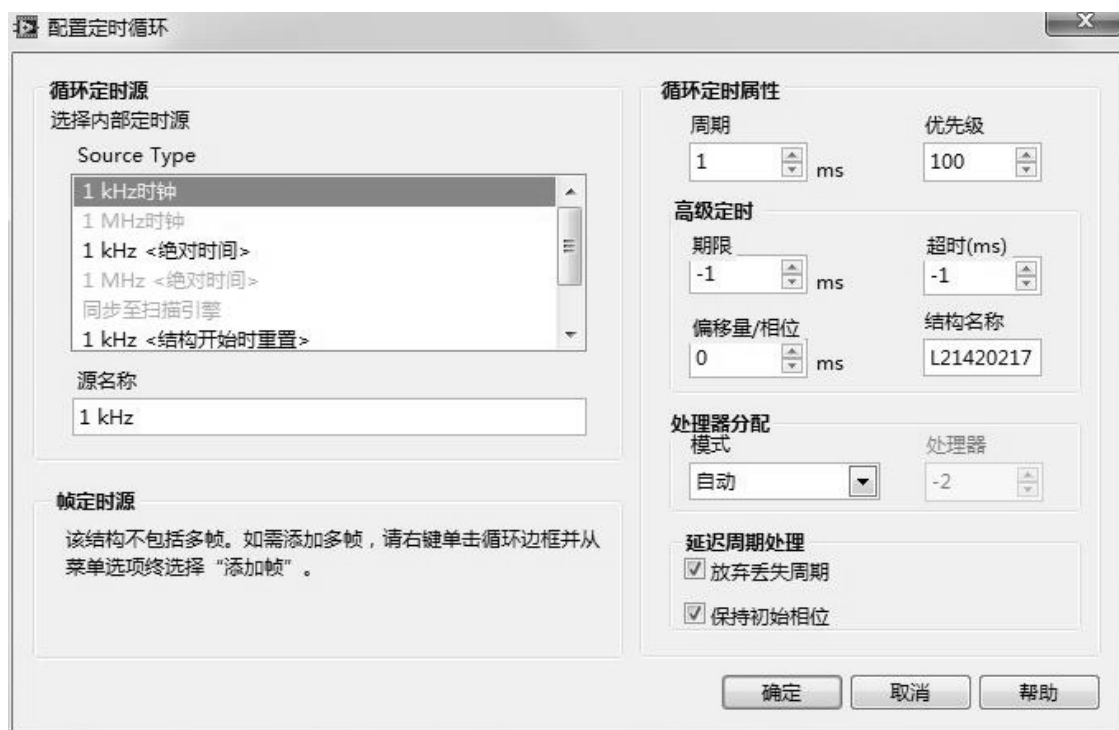


图 6.57 设置定时循环

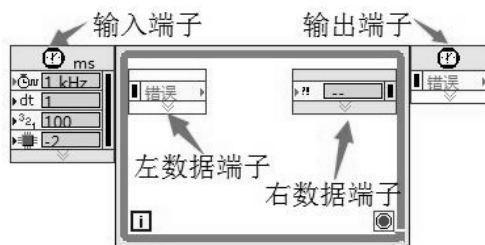


图 6.58 定时循环结构

定时循环的左侧数据节点用于返回各配置参数值并提供上一次循环的定时和状态信息，如循环是否延迟执行、循环实际起始执行时间和循环的预计执行时间等。可以将各值连接至右数据端子的输入端，以动态配置下一次循环，或右键单击右侧数据节点，在快捷菜单中选择“配置输入节点”的“配置下一次循环”对话框，输入各参数值。

输出端子返回由输入节点错误输入端输入的信息、执行中结构产生的错误信息，或在定

时循环内执行的任务子程序框图所产生的错误信息，输出端子还返回定时和状态信息。

输入端子的下侧有 6 个可能的端口，用鼠标左键附在输入端口可以看到其各自的名称，包括定时源、周期、优先级、期限、名称和模式。

- 定时源决定了循环能够执行的最高频率，默认为 1kHz。
- 周期为相邻两次循环之间的时间间隔，其单位由定时源决定。当采用默认定时源时，循环周期的单位为毫秒。优先级为整数，数字越大，优先级越高。
- 优先级的概念是在同一程序框图中的多个定时循环之间相对而言的，即在其他条件相同的前提下，优先级高的定时循环先被执行。
- 名称是定时循环的一个标志，一般被作为停止定时循环的输入参数，或者用来标识具有相同的启动时间的定时循环组。
- 执行定时循环的某一次循环的时间可能比指定的时间晚，模式决定了如何处理这些迟到的循环，处理方式如下所述。
  - (1) 定时循环调度器可以继续已经定义好的调度计划。
  - (2) 定时循环调度器可以定义新的执行计划，并且立即启动。
  - (3) 定时循环可以处理或丢弃循环。

当向定时循环添加帧时，可顺序执行多个子程序框图并指定循环中每次循环的周期，形成了一个多帧定时循环，如图 6.59 所示。多帧定时循环相当于一个带有嵌入顺序结构的定时循环。

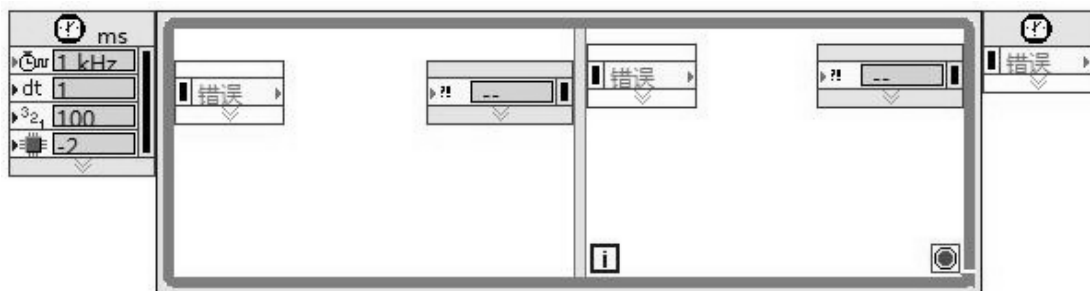


图 6.59 多帧定时循环

定时顺序结构由一个或多个任务子程序框图或帧组成，是根据外部或内部信号时间源定时后顺序执行的结果。定时顺序结构适用于开发精确定时、执行反馈、定时特征等动态改变或有多层执行优先级的 VI。定时顺序结构如图 6.60 所示。

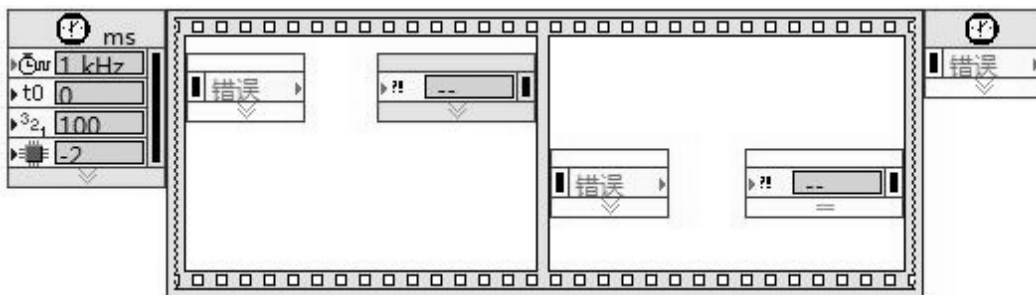


图 6.60 定时顺序结构

## 6.6.2 配置定时循环和定时顺序结构

配置定时循环主要包括以下几个方面。

### 1. 配置下一帧

双击当前帧的右侧数据节点或右键单击该节点，从快捷菜单中选择“配置输入节点”，打开“配置下一次循环”对话框，如图 6.61 所示。



图 6.61 “配置下一次循环”对话框

在这个对话框中，可为下一帧设置优先级、执行期限以及超时等选项。开始时间指定了下一帧开始执行的时间。要指定一个相对于当前帧的起始时间值，其单位应与帧定时源的绝对单位一致。在开始文本框中指定起始时间值，还可使用帧的右侧数据节点的输入端动态配置下一次定时循环或动态配置下一帧。在默认状态下，定时循环帧的右侧数据节点不显示所有可用的输出端。如果需要显示所有可用的输出端，可调整右侧数据节点大小或右键单击右侧数据节点，并从快捷菜单中选择显示隐藏的接线端。

### 2. 设置定时循环周期

周期指定各次循环间的时间长度，以定时源的绝对单位为单位。

图 6.62 所示程序框图的定时循环使用默认的 1kHz 定时源。循环 1 的周期为 1000ms，循环 2 的周期为 2000ms，这意味着循环 1 执行一次的时间为 1s，循环 2 执行一次的时间为 2s。这两个定时循环均在 6 次循环后停止执行。循环 1 于 6s 后停止执行，循环 2 则在 12s 后

停止执行。

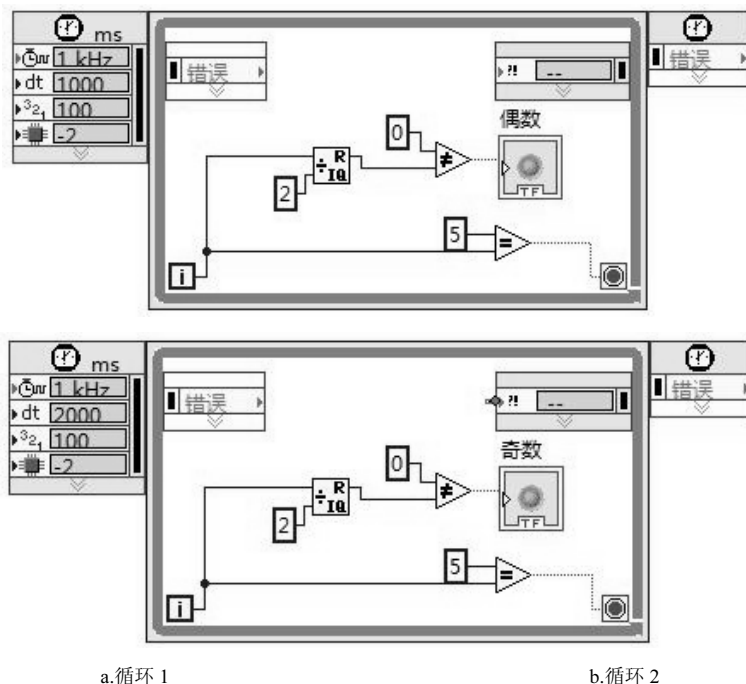


图 6.62 定时循环的简单使用

### 3. 设置定时结构的优先级

定时结构的优先级指定了定时结构相对于程序框图上其他对象开始执行的时间。设置定时结构的优先级，可使应用程序中存在多个在同一 VI 中互相预占执行顺序的任务。定时结构的优先级越高，它相对于程序框图中其他定时结构的优先级便越高。优先级的输入值必须为 1~255 之间的正整数。

程序框图中的每个定时结构会创建和运行含有单一线程的自有执行系统，因此不会出现并行的任务。定时循环的执行优先级介于实时和高之间。这意味着在一个程序框图的数据流中，定时循环总是在优先级不是实时的 VI 前执行。

所以，如同前面所介绍，若程序框图中同时存在优先级设为实时的 VI 和定时顺序，将导致无法预计的定时行为出现。

用户可为每个定时顺序或定时循环的帧指定优先级。运行包含定时结构的 VI 时，LabVIEW 将检查结构框图中所有可执行帧的优先级，并从优先级实时的帧开始执行。

使用定时循环时，可将一个值连接至循环最后一帧的右侧数据节点的“优先级”输入端，以动态设置定时循环后续各次循环的优先级。对于定时结构，可将一个值连接至当前帧的右侧数据节点，以动态设置下一帧的优先级。在默认状态下，帧的右侧数据节点不显示所有可用的输出端。如果需要显示所有可用的输出端，可以调整右侧数据节点的大小或右键单击右侧数据节点，并从快捷菜单中选择显示隐藏的接线端。

如图 6.63 所示, 程序框图包含了一个定时循环及定时顺序。定时顺序第一帧的优先级 (100) 高于定时循环的优先级 (100), 因此定时顺序的第一帧先执行。定时顺序第一帧执行完毕后, LabVIEW 将比较其他可执行的结构或帧的优先级。定时循环的优先级 (100) 高于定时顺序第二帧 (50) 的优先级。LabVIEW 将执行一次定时循环, 再比较其他可执行的结构或帧的优先级。定时循环的优先级 (100) 高于定时顺序第二帧的优先级 (50)。在本例中, 定时循环将在定时顺序第二帧执行前完全执行完毕。

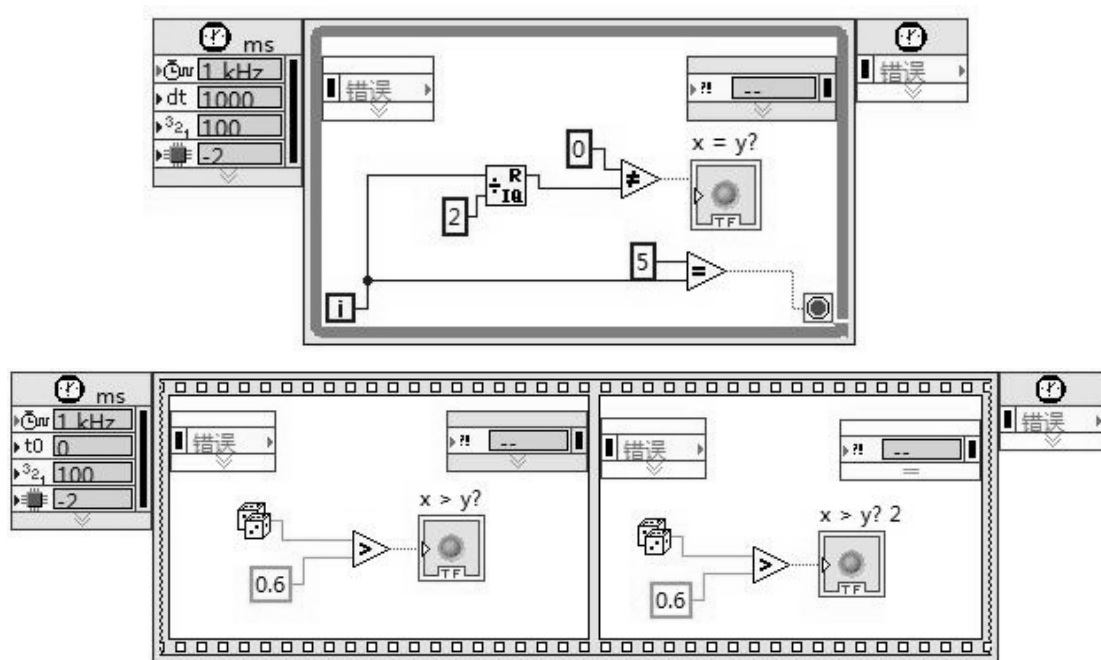


图 6.63 定时循环的优先级设置

#### 4. 选择定时结构的定时源

定时源控制着定时结构的执行, 有内部或外部两种定时源可供选择。内部定时源可在定时结构输入节点的配置对话框中选择。外部定时源可通过创建定时源 VI 及 DAQmx 中的数据采集 VI 来创建。内部定时源用于控制定时结构的内部定时, 包括操作系统自带的 1kHz 时钟及实时 (RT) 终端的 1MHz 时钟。通过配置定时循环、配置定时顺序或配置多帧定时循环对话框的循环定时源或顺序定时源, 可选中一个内部定时源。

- 1kHz 时钟: 在默认状态下, 定时结构以操作系统的 1kHz 时钟为定时源。如果使用 1kHz 时钟, 定时结构每毫秒执行一次循环。所有可运行定时结构的 LabVIEW 平台都支持 1kHz 定时源。
- 1MHz 时钟: 终端可以使用终端处理器的 1MHz 时钟来控制定时结构。如果使用 1MHz 时钟, 定时结构每微秒执行一次循环。如终端没有系统所支持的处理器, 便不能选择使用 1MHz 时钟。
- 1kHz 时钟<结构开始时重置>: 与 1kHz 时钟相似的定时源, 每次定时结构循环后重置为 0。

- 1MHz 时钟<结构开始时重置>：与 1MHz 时钟相似的定时源，每次定时结构循环后重置为 0。

外部定时源用于创建控制定时结构的外部定时。使用创建定时源 VI 通过编程选中一个外部定时源。另有几种类型的 DAQmx 定时源可用于控制定时结构，如频率、数字边缘计数器、数字改动检测和任务源生成的信号等。通过 DAQmx 的数据采集 VI 可创建用于控制定时结构的 DAQmx 定时源。可使用次要定时源控制定时结构中各帧的执行。例如，以 1kHz 时钟控制定时循环，以 1MHz 时钟控制每次循环中各帧的定时。

### 5. 设置执行期限

执行期限是指执行一个子程序框图或一帧所需要的时间。执行期限与帧的起始时间相对。通过执行期限可设置子程序框图的时限。如子程序框图未能在执行期限前完成，下一帧的左侧数据节点将在“延迟完成？”输出端返回真值并继续执行。指定一个执行期限，其单位与帧定时源的单位一致。

在如图 6.64 中，定时顺序中首帧的执行期限已配置为 50。执行期限指定子程序框图要在 1kHz 时钟走满 50 下前结束执行，即在 50 ms 前完成。而子程序框图耗时 100 ms 完成代码执行。当帧无法在指定的最后期限前结束执行代码时，第二帧的“延迟完成？”输出端将返回真值。

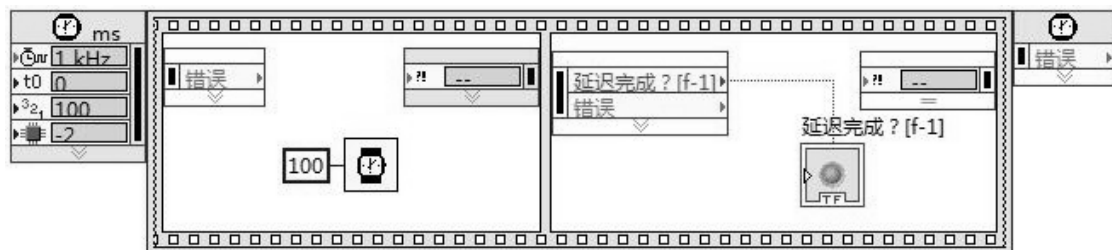


图 6.64 设置执行期限

### 6. 设置超时

超时是指子程序框图开始执行前可等待的最长时间，以毫秒为单位。超时与循环起始时间或上一帧的结束时间相对。如果子程序框图未能在指定的超时前开始执行，定时循环将在该帧的左侧数据节点的“唤醒原因”输出端中返回超时，如图 6.65 所示，定时顺序的第一帧耗时 50ms 执行，第二帧配置为定时顺序开始 51ms 后再执行。第二帧的超时设为 10ms，这意味着，该帧将在第一帧执行完毕后等待 10ms 再开始执行。如第二帧未能在 10ms 前开始执行，定时结构将继续执行余下的非定时循环，而第二帧则在左侧数据节点的“唤醒原因”输出端中返回超时。

余下各帧的定时信息与发生超时帧的定时信息相同。如果定时循环必须再完成一次循环，则循环会停止于发生超时的帧，等待最初的超时事件。定时结构第一帧的超时默认值为 -1，即无限等待子程序框图或帧的开始。其他帧的超时默认值为 0，即保持上一帧的超时

值不变。

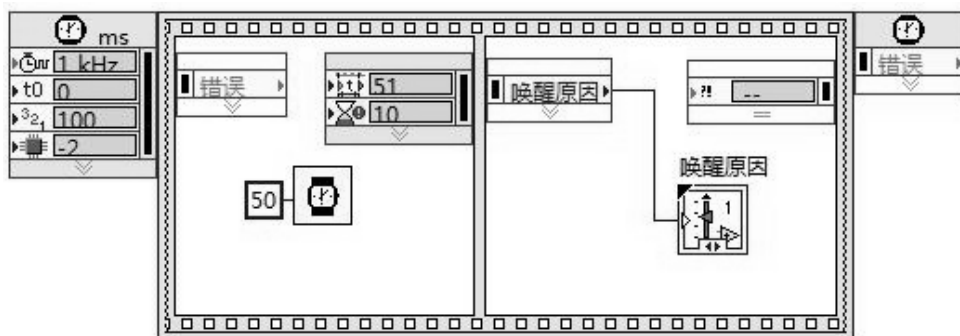


图 6.65 设置超时

## 7. 设置偏移

偏移是相对于定时结构开始时间的长度，这种结构等待第一个子程序框图或帧执行的开始。偏移的单位与结构定时源的单位一致。还可以在不同定时结构中使用与定时源相同的偏移对齐不同定时结构的相位，如图 6.66 所示，定时循环都使用相同的 1kHz 定时源，且偏移值（t0）为 500，这意味着循环将在定时源触发循环开始后等待 500ms。

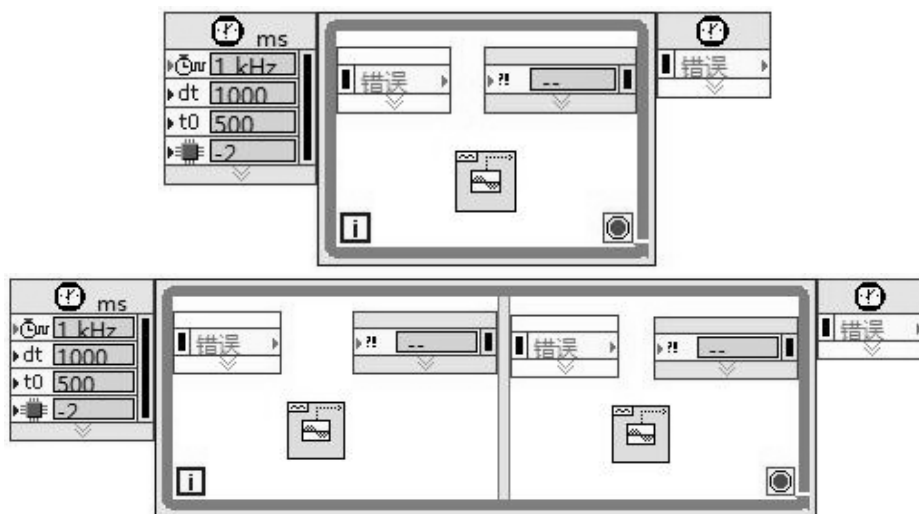


图 6.66 设置偏移

在定时循环的最后一帧中，可使用右侧数据节点动态改变下一次循环的偏移。然而，在动态改变下一次循环的偏移时，必须将值连接至右侧数据节点的模式输入端，以指定一个模式。

如果通过右侧数据节点改变偏移，则必须选择一个模式值。

对齐两个定时结构无法保证二者的执行开始时间相同。使用同步定时结构起始时间，可以令定时结构的执行起始时间同步。



## 6.7 变 量

“结构”选板中的变量包含共享变量、局部变量和全局变量，变量在“结构”选板中的位置如图 6.67 所示。



图 6.67 变量在“结构”选板中的位置

局部和全局变量是高级的 LabVIEW 概念，它们不是 LabVIEW 数据流执行模型中固有的部分。使用局部变量和全局变量时，程序框图可能会变得难以阅读。所以，对于全局变量来说，同使用局部变量一样，需要谨慎。错误地使用局部变量和全局变量，如将其取代连线板或用于访问顺序结构中每一帧的数值，可能在 VI 中出现不可预期的行为。滥用局部变量、全局变量，如用来避免程序框图间的过长连线或取代数据流，将会降低执行速度。需要注意的是，对于局部变量和全局变量，应确保 VI 运行前局部变量和全局变量含有已知的数据值。否则变量可能含有导致 VI 发生错误行为的数据。若在 VI 第一次读取变量之前没有将变量初始化，则变量含有的是相关前面板对象的默认值。

## 6.7.1 局部变量

创建局部变量的方法有两种：第一种方法是直接在程序框图中已有的对象上单击鼠标右键，从弹出的快捷菜单中创建局部变量，如图 6.68 所示。

第二种方法是在函数选板中的“结构”子选板中选择局部变量，形成一个没有被赋值的变量，此时的局部变量没有任何用处，因为它还没有和前面板的控制或指示相关联，这时可以通过在前面板添加控件来填充其内容，如图 6.69 所示。

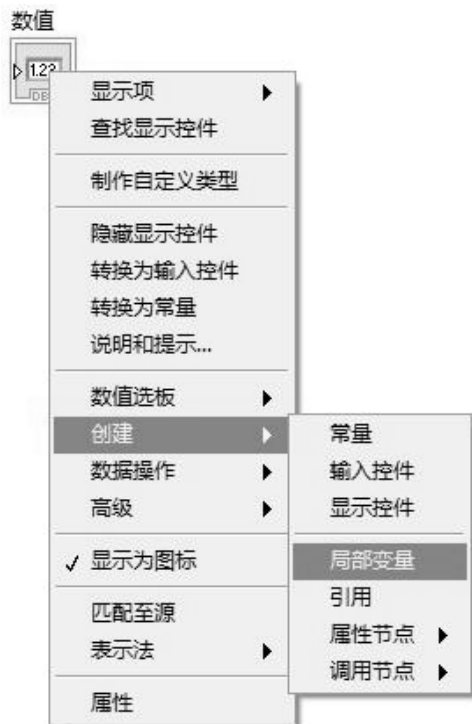


图 6.68 创建局部变量方法一



图 6.69 创建局部变量方法二

使用局部变量可以在一个程序的多个位置实现对前面板控件的访问，也可以在无法连线的框图区域之间传递数据。每一个局部变量都是对某一个前面板控件数据的引用。可以为一个输入量或输出量建立任意多的局部变量，从它们中的任何一个都可以读取控件中的数据，向这些局部变量中的任何一个写入数据，都将改变控件本身和其他局部变量。

图 6.70 显示了使用同一个开关同时控制两个 While 循环。

一个局部变量就是其对应前面板对象的一个复制，要占一定的内存，所以使用过多的局部变量会占用大量内存，尤其当局部变量是数组这样的复合数据类型时。所以在使用局部变量时要先考虑内存，局部变量会复制数据缓冲区。从一个局部变量读取数据时，便为相关控件的数据创建了一个新的缓冲区。如果使用局部变量将大量数据从程序框图上的某个地方传到另一个地方，通常会使用更多的内存，最终导致执行速度比使用连线来传递数据更慢。若在执行期间需要存储数据，可考虑使用移位寄存器。过多使用局部变量还会使程序的可读性变差，并且有可能导致不易发现的错误出现。

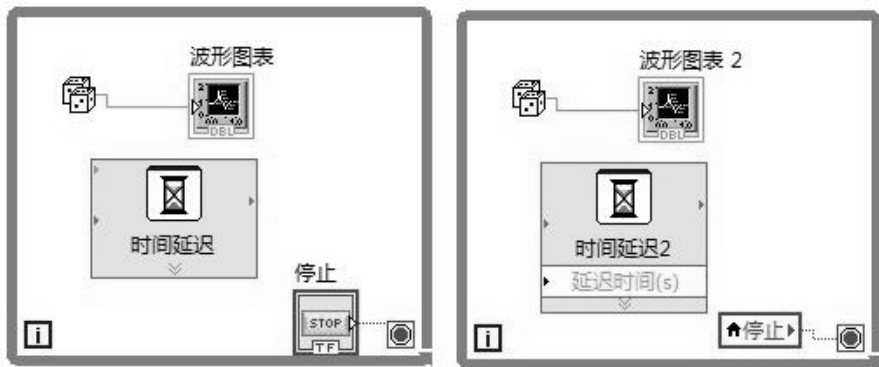


图 6.70 同时控制两个 While 循环的程序框图

局部变量还可能引起竞态问题，如图 6.71 所示，此时无法估计数值的最终值是多少，因为无法确认两个并行执行代码在时间上的执行顺序。存在竞态问题程序的输出取决于各运算的运行顺序。由于这两个运算间没有数据依赖关系，因此很难判断哪个运算先运行。为避免竞争状态，可以使用数据流或顺序结构，以强制加入运行顺序控制的机制，或者不要同时读写同一个变量。



图 6.71 竞态问题举例

局部变量只能在同一个 VI 中使用，而不能在不同的 VI 之间使用。若需要在不同的 VI 之间进行数据传递，则要使用全局变量。

### 6.7.2 全局变量

全局变量的创建也有两种方法：第一种是在结构的子选板中，从中选择全局变量，生成一个小图标，双击该图标，弹出框图，如图 6.72 所示。在框图内可编辑全局变量。

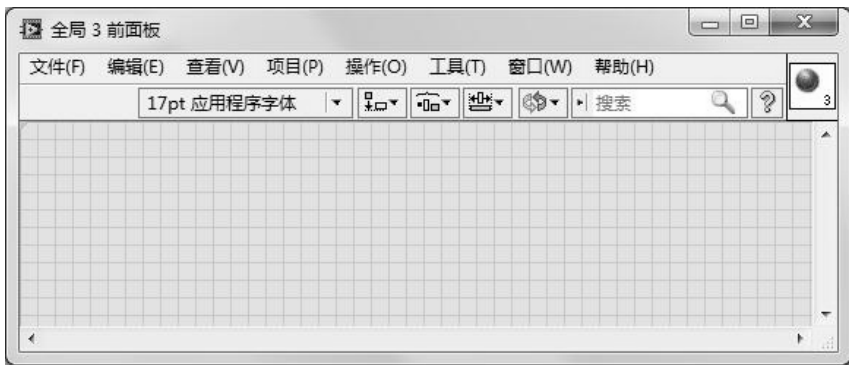


图 6.72 创建全局变量方法一

第二种方法是在 LabVIEW 中的新建菜单中选择全局变量，如图 6.73 所示。单击“确定”按钮后就可以打开设计全局变量窗口。

但此时只是一个没有程序框图的 LabVIEW 程序，要使用全局变量可按照以下步骤进行：

第一步，向刚才的前面板内添加想要的全局变量，如想添加的数据。

第二步，保存这个全局变量，然后关闭全局变量的前面板窗口。

第三步，新建一个程序，打开其程序框图，从函数选板中选择“选择 VI”，打开保存的文件，拖出一个全局变量。

第四步，右键单击图标，从弹出的菜单中选择“选择项”，就可以根据需要选择相应的变量了。



图 6.73 创建全局变量方法二

全局变量可以同时运行的几个 VI 之间传递数据，例如，可以在一个 VI 中向全局变量写入数据，在随后运行的同一程序中的另一个 VI 中从全局变量读取写好的数据。通过全局变量在不同的 VI 之间进行数据交换只是 LabVIEW 中 VI 之间数据交换的方式之一，通过动态数据交换也可以进行数据交换。需要注意的是，在一般情况下，不能利用全局变量在两个 VI 之间传递实时数据，原因是通常情况下两个 VI 对全局变量的读写速度不能保证严格一致。

# 第7章 数组/字符串/簇和矩阵

## 7.1 数 组

在程序设计语言中，数组（Array）是一种常用的数据结构，是相同数据类型数据的集合，是一种存储和组织相同类型数据的良好方式。与其他程序设计语言一样，LabVIEW 中的数组是数值型、布尔型、字符串型等多种数据类型中的同类数据的集合，在前面板的数组对象通常由一个盛放数据的容器和数据本身构成，在程序框图上则体现为一个一维或多维矩阵，数组在函数选板和控件选板中的位置如图 7.1 所示。

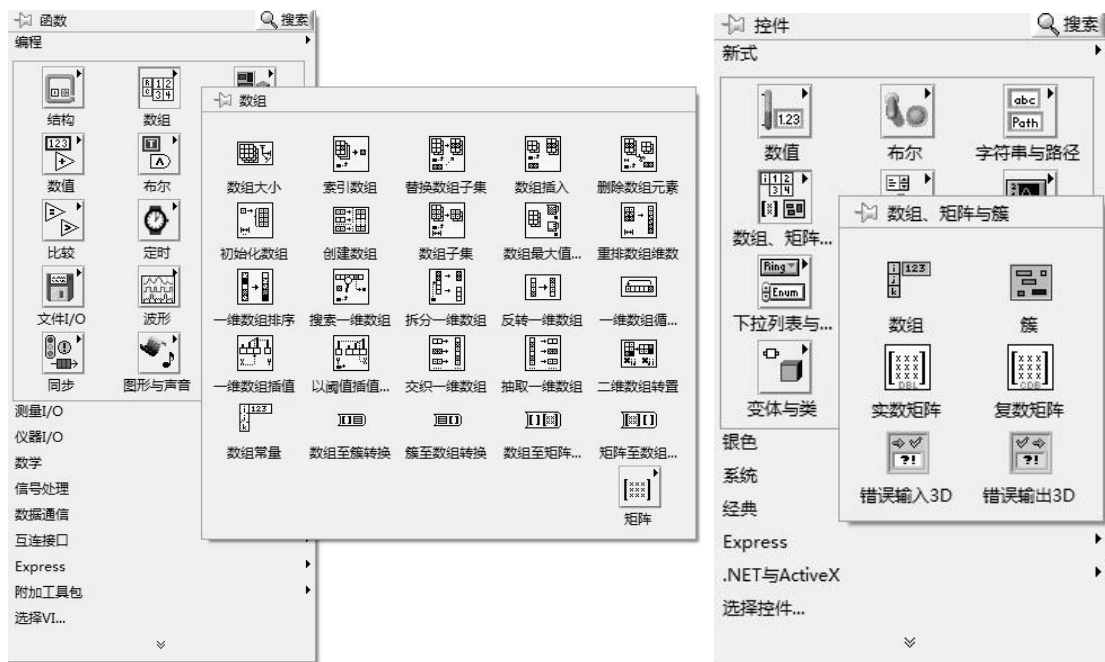


图 7.1 数组的位置

数组中的每一个元素都有其唯一的索引值，可以通过索引值来访问数组中的数据。下面详细介绍数组数据以及处理数组数据的方法。

### 7.1.1 数组的组成与创建

数组是由同一类型数据元素组成的大小可变的集合。当有一串数据需要处理时，它们可能是一个数组，当需要频繁地对一批数据进行绘图时，使用数组将会受益匪浅，数组作为组

织绘图数据的一种机制也是十分有用的。当执行重复计算，或解决能自然描述成矩阵向量符号的问题时数组也是很有用的，如解答线性方程。在 VI 中使用数组能够压缩框图代码，并且由于具有大量的内部数组函数和 VI，使得代码开发更加容易。

可通过以下两步来实现数组输入控件或数组显示控件的创建：

(1) 从控件选板中选取数组、矩阵控件，再选其中的数组拖入前面板中。

(2) 将需要有效数据对象拖入数组框，切记此要点，如果不分配数据类型，该数组将显示为带空括号的黑框。

在数组框图的左端或右上角为数组的索引值，显示在数组左边方框中的索引值对应数组中第一个可显示的元素，通过索引值的组合可以访问数组中的每一个元素。LabVIEW 中的数组与其他编程语言相比比较灵活，任何一种数据类型的数据（数组本身除外）都可以组成数组。其他的编程语言如 C 语言，在使用一个数组时，必须首先定义数组的长度，但 LabVIEW 却不必如此，它会自动确定数组的长度。在内存允许的情况下，数组中每一维的元素最多可达 230 个。数组中元素的数据类型必须完全相同，如都是无符号 16 位整数，或全为布尔型等。当数组中有  $n$  个元素时，元素的索引号从 0 开始，到  $n-1$  结束。

### 7.1.2 使用循环创建数组

数组经常要用一个循环来创建，其中 For 循环是最合适的，这是因为 For 循环的循环次数是预先指定的，在循环开始前它已分配好了内存，而 While 循环却无法做到这一点，因为无法预先知道 While 循环将循环多少次。

对于 For 循环来说，默认状态下是允许自动索引的。但对于 While 循环，默认状态下自动索引被禁用。若希望能够自动索引，需要从 While 循环上弹出的快捷菜单中选择启用索引。当不知道数组的具体长度时，使用 While 循环是最合适的，用户可以根据需要设定循环终止条件。

### 7.1.3 数组函数

对于一个数组可进行很多操作，比如求数组的长度、对数组进行排序、查找数组中的某一元素、替换数组中的元素等。传统的编程语言主要依靠各种数组函数来实现这些运算，而在 LabVIEW 中，这些函数是以功能函数节点的形式来表现的。

下面将介绍几种常用的数组函数。

#### 1) 数组大小 (Array Size)

数组大小的连线板如图 7.2 所示。

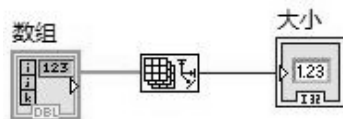


图 7.2 数组大小的连线板

数组大小函数返回输入数组的元素个数，节点的输入为一个  $n$  维数组，输出为该数组各

维包含元素的个数。当  $n=1$  时，节点的输出为一个标量。当  $n>1$  时，节点的输出为一个一维数组，数组的每个元素对应输入数组中每一维的长度。

## 2) 创建数组 (Build Array)

创建数组的连线板如图 7.3 所示。



图 7.3 创建数组的连线板

创建数组函数用于合并多个数组或给数组添加元素。函数有两种类型的输入：标量和数组，因此函数可以接受数组和单值元素输入，节点将从左侧端口输入的元素或数组按从上到下的顺序组成一个新数组。

当两个数组需要连接时，可以将数组看成整体，即作为一个元素。有时可能根据需要使用创建数组函数时，不是将两个一维数组合成一个二维数组，而是将两个一维数组连接成一个更长的一维数组；或者不是将两个二维数组连接成一个三维数组，而是将两个二维数组连接成一个新的二维数组。在这种情况下，需要利用创建数组节点的连接输入功能，在创建数组节点的右键快捷菜单中选择“连接输入”命令，创建数组的图标也有所改变。

## 3) 一维数组排序 (Sort 1D Array)

一维数组排序的连线板如图 7.4 所示。



图 7.4 一维数组排序的连线板

一维数组排序函数可以对输入的数组进行按升序排序，若用户想按降序排序，可以与反转一维数组函数组合，实现对数组的降序排列。

## 4) 索引数组 (Index Array)

索引数组的连线板如图 7.5 所示。

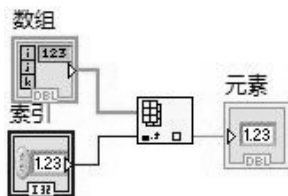


图 7.5 索引数组的连线板

索引数组用于访问数组的一个元素，使用输入索引指定要访问的数组元素，第  $n$  个元素的索引号是  $n-1$ 。

索引数组函数会自动调整大小以匹配连接的输入数组维数，若将一维数组连接到索引函数，函数将显示一个索引输入，若将二维数组连接到索引函数，将显示两个索引输入，即索引（行）和索引（列），当索引输入仅连接行输入时，则抽取完整一维数组的那一行；若仅连接列输入时，那么将抽取完整的一维数组的那一列；若连接了行输入和列输入，那么将抽取数组的单个元素。每个输入数组是独立的，可以访问任意维数组的任意部分。

### 5) 初始化数组 (Initialize Array)

初始化数组的连线板如图 7.6 所示。



图 7.6 初始化数组的连线板

初始化数组函数的功能是为了创建  $n$  维数组，数组维数由函数左侧的维数大小端口的个数决定。创建之后每个元素的值都与输入到元素端口的值相同。函数刚放在程序框图上时，只有一个维数大小输入端子，创建的是指定大小的一维数组。此时可以通过拖动下边缘或在维数大小端口的右键选单中选择添加维度，以添加维数大小端口。

在 LabVIEW 中初始化数组还有其他方法。若数组中的元素都是相同的，用一个带有常数的 For 循环即可初始化，这种方法的缺点是创建数组时要占用一定的时间。

若元素值可以由一些直接的方法计算出来，把公式放到前一种方法中的 For 循环中取代其常数即可。例如，这种方法可以产生一个特殊波形。也可以在框图程序中创建一个数组常量，手动输入各个元素的数值，而后将其连接到需要初始化的数组上。这种方法的缺点是烦琐，并且在存盘时会占用一定的磁盘空间。如果初始化数组所用的数据量很大，可以先将其放到一个文件中，在程序开始时再装载。

需要注意的是，在初始化时有一种特殊情况，那就是空数组，空数组不是一个元素值为 0、假、空字符串或类似的数组，而是一个包含零个元素的数组，相当于 V 语言中创建了一个指向数组的指针。经常用到空数组的例子是初始化一个连有数组的循环移位寄存器。有以下几种方法创建一个空数组：用一个数组大小输入端口不连接数值或输入值为 0 的初始化函数来创建一个空数组；创建一个  $n$  为 0 的 For 循环，在 For 循环中放入所需数据类型的常量。For 循环将执行零次，在其框架通道上将产生一个相应类型的空数组，但是不能用创建数组函数来创建空数组，因为它的输出至少包含一个元素。

### 6) 替换数组子集 (Replace Array Subset)

替换数组子集函数是从新元素/子数组端口中输入，替换其中一个或部分元素。新元素/子数组输入的数据类型必须与输入数组的数据类型一致。



替换数组子集的连线板如图 7.7 所示。

### 7) 删除数组元素 (Delete From Array)

删除数组元素的连线板如图 7.8 所示。



图 7.7 替换数组子集的连线板



图 7.8 删除数组元素的连线板

删除数组元素函数用于从数组中删除指定数目的元素，索引端口用于指定所删除元素的起始元素的索引号，长度端口用于指定删除元素的数目。

## 7.2 簇

“簇”是 LabVIEW 中一种特殊的数据类型，是由不同数据类型数据构成的集合。在使用 LabVIEW 编写程序的过程中，不仅需要相同数据类型的集合——数组来进行数据的组织，有些时候也需要将不同数据类型的数据组合起来以更加有效地行使其功能。

### 7.2.1 簇的组成与创建

簇可以将几种不同的数据类型集中到一个单元中形成一个整体，类似于 C 语言中的结构。

簇通常用于将出现在框图上的有关数据元素分组管理。因为簇在框图中仅用唯一的连线表示，所以可以减少连线混乱和子 VI 需要的连接器端子个数。使用簇有积极的效果，可以将簇看作一捆连线，其中每条连线表示簇不同的元素。在框图上，只有当簇具有相同元素类型、相同元素数量和相同元素顺序时，才可以将簇的端子连接。

簇和数组的异同：簇可以包含不同类型的数据，而数组仅可以包含相同的数据类型，簇和数组中的元素都是有序排列的，但访问簇中元素最好是通过释放方法同时访问其中的部分或全部元素，而不是通过索引一次访问一个元素。簇和数组的另一个差别是簇具有固定的大小。簇和数组的相似之处是二者都是由输入控件或输出控件组成的，不能同时包含输入控件和输出控件。

一个簇变为输入控件簇或显示控件簇取决于放进簇中的第一个元素，若放进簇框架中的第一个元素是布尔控件，那么后来给簇添加的任何元素都将变成输入对象，簇变为输入控件簇，并且当从任何簇元素的快捷菜单中选择转换为输入控件或转换为显示控件时，簇中的所有元素都将发生变化。

在簇框架上单击右键弹出快捷菜单，在菜单“自动调整大小”中的三个选项可以用来调整簇框架的大小以及簇元素的布局，调整为匹配大小选项，调整簇框架的大小以适合所包含的所有元素；水平排列选项水平压缩排列所有元素；垂直排列选项垂直压缩排列所有元素。

簇的元素有一定的顺序，簇元素按照它们放入簇中的先后顺序排序，而不是按照簇框架内的物理顺序排序，簇框架中的第一个对象标记为 0，第二个为 1，依次排列。在簇中删除元素时，剩余元素的顺序将自行调整，在簇的解除捆绑和捆绑函数中，簇顺序决定了元素的显示顺序。如果要访问簇中的单个元素，必须记住簇顺序，因为簇中的单个元素是按顺序访问的。

在前面板上，从簇边框上右键快捷菜单中选择“重新排序簇中控件”，可以检查和改变簇内元素的顺序，此时图中的工具变成了一组新按钮，簇的背景也有变化，连光标也改变为簇排序光标，选择“重新排序簇中控件”后，簇中每一个元素右下角出现了并排的框，白框和黑框，白框指出该元素在簇顺序中的当前位置，黑框指出在用户改变顺序后的新位置，在此顺序改变前，白框和黑框中的数字是一样的，用簇排序光标单击某个元素，该元素在簇顺序中的位置就会变成顶部工具条显示的数字，单击“×”按钮后可恢复以前的排列顺序。

应注意簇顺序的重要性，使用簇时应当遵循的原则是：在一个高度交互的面板中，不要把一个簇既作为输入又作为输出。

## 7.2.2 簇数据的使用

对簇数据进行处理函数位于“函数选板→编程→簇与变体”子选板，如图 7.9 所示。

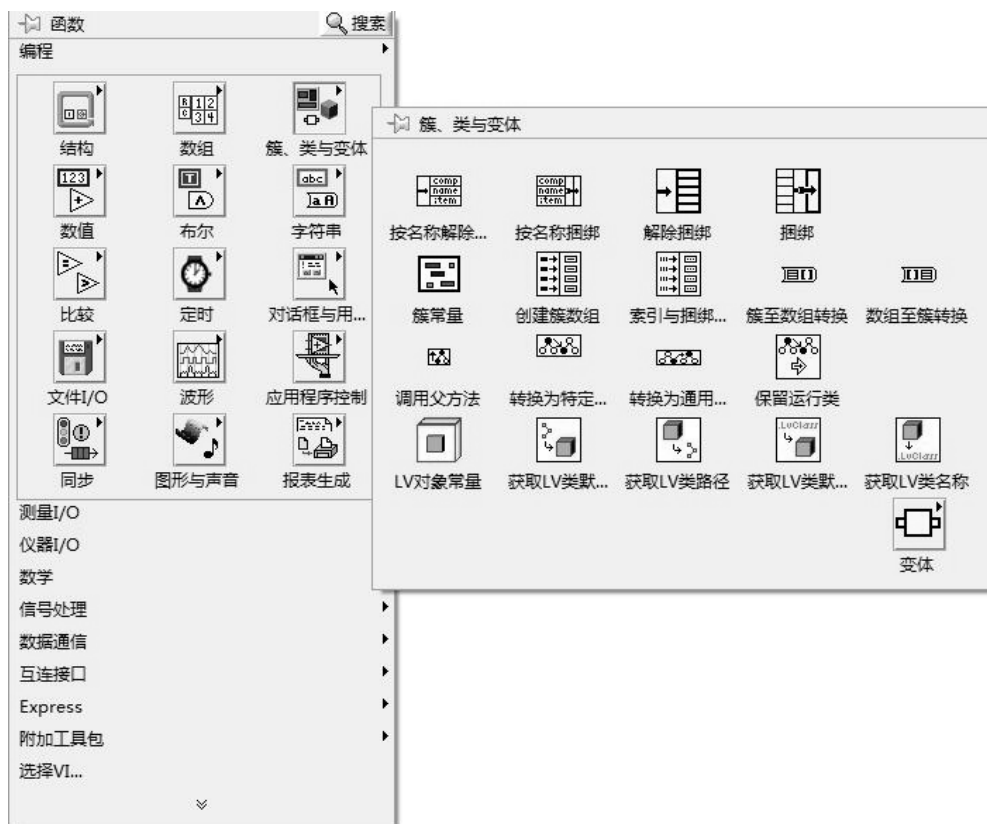


图 7.9 簇在函数选板中的位置

### 1) 解除捆绑 (Unbundle)

解除捆绑的连线板如图 7.10 所示。

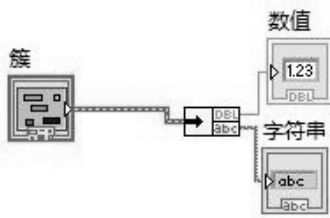


图 7.10 解除捆绑的连线板

解除捆绑函数用于从簇中提取单个元素，并将解除后的数据成员作为函数的输出结果。当解除捆绑未接入输入参数时，右端只有两个输出端口，当接入一个簇时，解除捆绑函数会自动检测输入簇的元素个数，生成相应个数的输出端口。

### 2) 按名称解除捆绑 (Unbundle By Name)

按名称解除捆绑的连线板如图 7.11 所示。

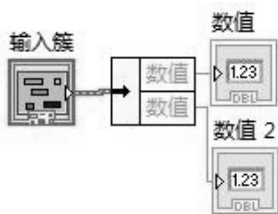


图 7.11 按名称解除捆绑的连线板

按名称解除捆绑是把簇中的元素按标签解除捆绑，只有对有标签的元素，按名称解除捆绑的输出端才能弹出带有标签的簇元素的标签列表。对没有标签的元素，输出端不弹出其标签列表，输出端口的个数不限，可以根据需要添加任意数目的端口。

### 3) 捆绑 (Bundle)

捆绑的连线板如图 7.12 所示。

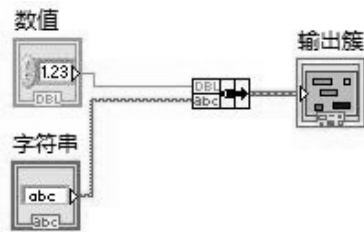


图 7.12 捆绑的连线板

捆绑函数用于将若干基本数据类型的数据元素合成为一个簇数据，也可以替换现有簇中的值，簇中元素的顺序和捆绑函数的输入顺序相同。顺序定义是从上到下，即连接顶部的元

素变为元素 0，连接到第二个端子的元素变为元素 1。

#### 4) 按名称捆绑 (Bundle By Name)

按名称捆绑的连线板如图 7.13 所示。

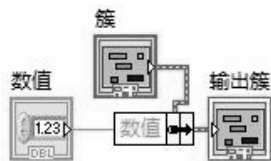


图 7.13 按名称捆绑的连线板

该函数功能与捆绑函数类似，但只用于将元素捆绑为簇，输入接线端“输入簇”必须被连接至一个已有的簇，且簇中的元素至少应有一个有标签名。

#### 5) 创建簇数组 (Build Cluster Array)

创建簇数组的连线板如图 7.14 所示。

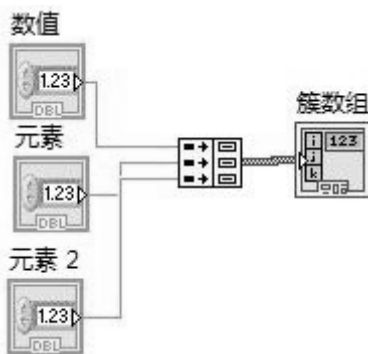


图 7.14 创建簇数组的连线板

创建簇数组函数的用法与创建数组函数的用法类似，与创建数组不同的是其输入端口的分量元素可以是簇。函数会首先将输入到输入端口的每个分量元素转化为簇，然后再将这些簇组成一个簇的数组，输入参数可以都为数组，但要求维数相同，要注意的是，所有从分量元素端口输入数据的类型必须相同，分量元素端口的数据类型与第一个连接进去的数据类型相同。

#### 6) 簇至数组转换 (Array to Cluster)

簇至数组转换的连线板如图 7.15 所示。



图 7.15 簇至数组转换的连线板

簇至数组转换函数要求输入簇的所有元素的数据类型必须相同，函数按照簇中元素的编号顺序将这些元素组成一个一维数组。

### 7) 数组至簇转换 (Cluster to Array)

数组至簇转换的连线板如图 7.16 所示。



图 7.16 数组至簇转换的连线板

数组至簇转换是簇至数组转换的逆过程。将数组转换为簇需要注意的是，此函数并不是将数组中所有的元素都转换为簇，而是将数组中的前  $n$  个元素组成一个簇， $n$  由用户自己设置，默认为 9，当  $n$  大于数组的长度时，函数会自动补充簇中的元素，元素值为默认值。

**例 7-1** 创建一个 VI，以软件名称、版本、是否已安装为内容建立一个簇，在簇中添加元素，形成新的簇，并提取新簇中的数据。

(1) 启动 LabVIEW 2013，打开前面板，选择控件选板中的“新式→数组、矩阵与簇→簇”，将簇控件放置于前面板。

(2) 选择“新式→字符串路径→字符串输入控件”、“新式→数组→数值显示控件”和“新式→布尔→圆形指示灯”，分别将其拖放在簇控件内，打包成一个簇，并分别为其赋值，如图 7.17 所示。



图 7.17 簇中元素

(3) 切换到程序框图，选择函数选板中的“编程→簇、类与变体→捆绑”函数，将其置于程序框图中，并调整输入端口为 3 个。将创建的 3 个控件分别连接到捆绑函数的输入端口，在捆绑函数的输出端口上单击鼠标右键，在弹出的快捷菜单中选择“创建→显示控件”命令，创建一个簇显示控件，并将其命名为软件信息，如图 7.18 所示。

(4) 在已创建好的“软件信息”簇中添加两个数据：发布时间和公司。在前面板中创建“发布时间”和“公司”两个字符串输入控件；在程序框图中添加一个捆绑函数并调整其输入端口为 3 个。将其与“发布时间”、“公司”和“软件信息”的输出端口连接，并在其输出

端口创建一个簇显示控件，命名为“详细信息”，如图 7.19 所示。

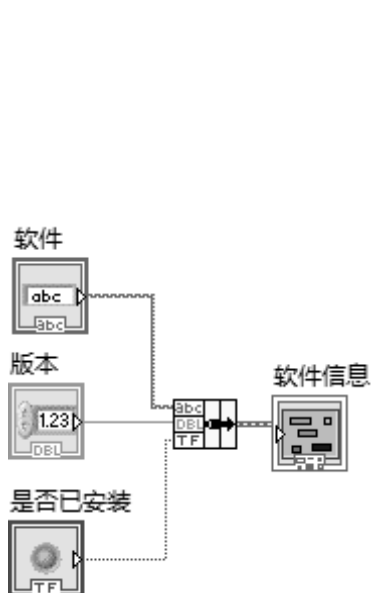


图 7.18 软件信息

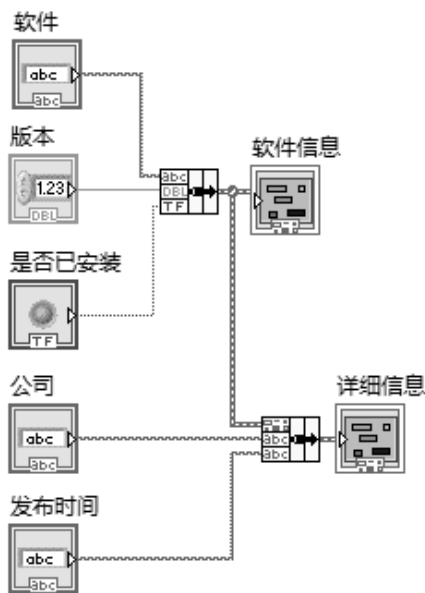


图 7.19 添加元素

(5) 运行程序，结果如图 7.20 所示。



图 7.20 添加元素后运行结果

(6) 选择“编程→簇、类与变体→按名称解绑”函数，将两个“按名称解绑”的输入端口分别连接到“软件信息”簇和“详细信息”簇，使用操作工具在第一个按名称解绑的函数上单击，选择“版本”选项，并创建一个“版本号显示”显示控件。用同样的方法在第二个按名称解绑的函数上创建“发布时间显示”显示控件，如图 7.21 所示。

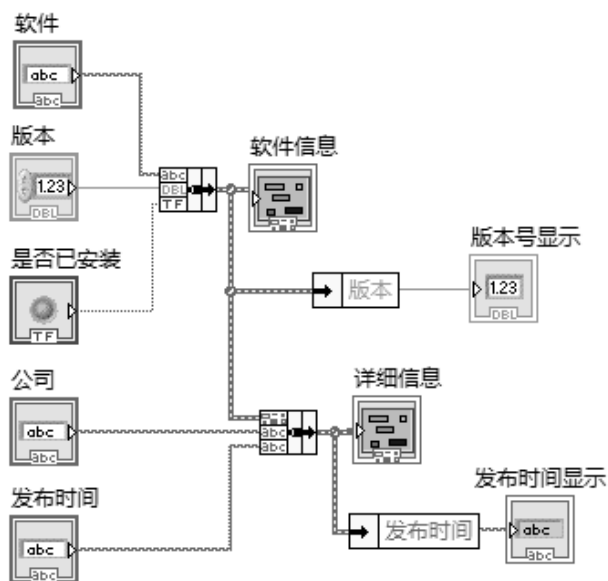


图 7.21 提取簇中元素

(7) 运行程序，完整的程序运行结果如图 7.22 所示。



图 7.22 完整的程序运行结果

## 7.3 字符串

字符串 (String) 在 LabVIEW 编程中会频繁被用到，因此 LabVIEW 封装了功能丰富的字符串函数，用于字符串的处理，用户不需要再像 C 语言中一样为字符串的操作编写烦琐的

程序。字符串在控制面板和函数选板中的位置如图 7.23 所示。

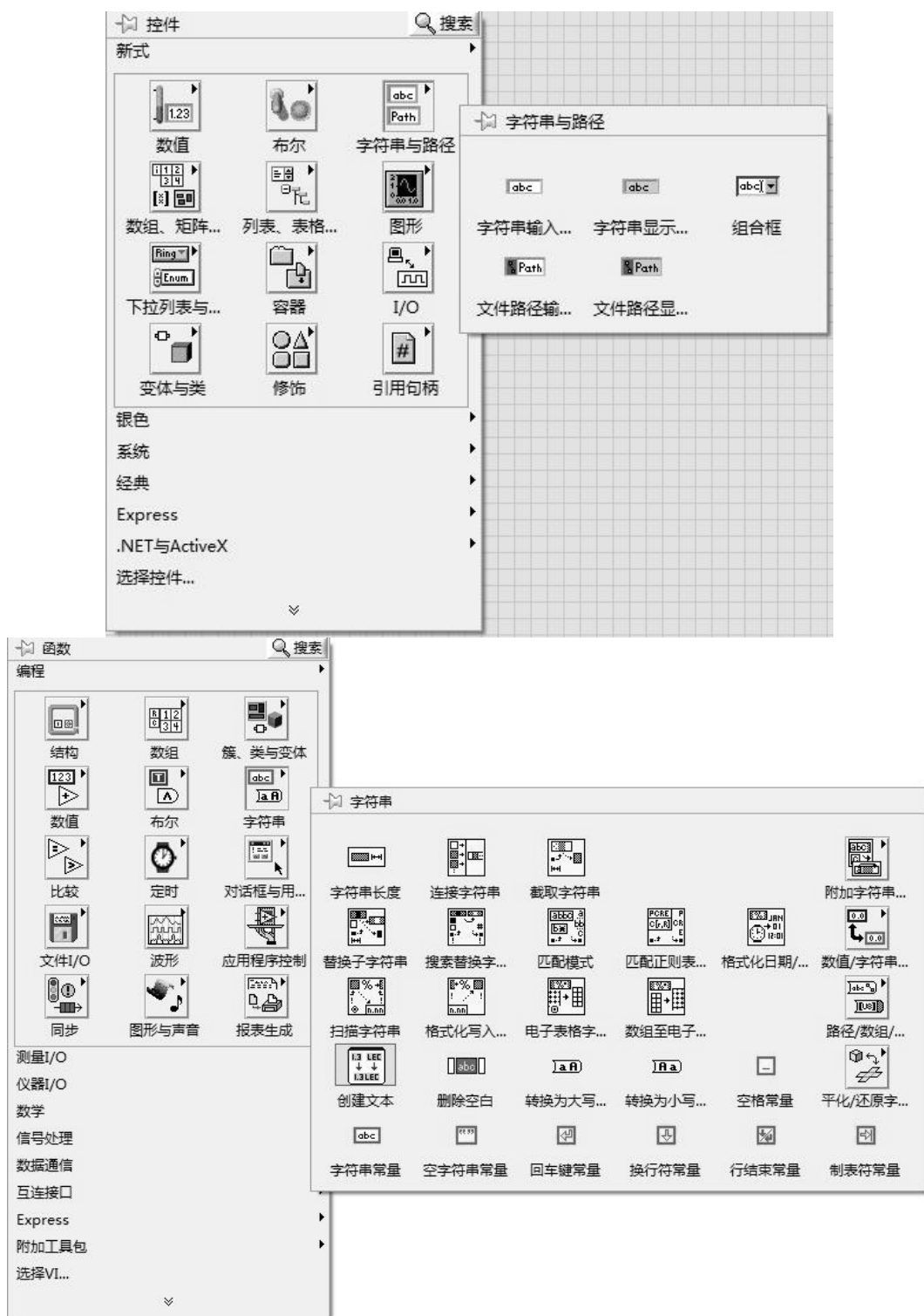


图 7.23 字符串在控制面板和函数选板中的位置



字符串相关控件包括输入控件、显示控件和下拉框。另外文件路径是 LabVIEW 中一种特殊的数据类型，方便用于文件的操作。但是由于它也兼具了字符串的特征，因此可以用一个很简单的 VI 函数实现它和字符串之间的转换。

### 7.3.1 字符串控件

字符串输入控件一般被用作文本输入框，而字符串显示控件一般被用作文本显示框。文本字体的颜色、大小和显示形式是可以改变的，如图 7.24 所示。选中字符串控件后，在修饰菜单下拉列表中就可以改变文字的字体、大小、颜色等，右击文本框可以选择文字的显示格式。



图 7.24 改变字符串的字体颜色等

下面介绍下拉框的用法。下拉框经常被用来当作选择菜单使用，如可以在登录框下再加一个“身份类型”下拉框。右击下拉框，选择 **Edit Items** 选项对下拉框的选项进行编辑。编辑完后，单击三角下拉箭头就可以看到下拉框中的内容了。下拉框的文本内容就是用户选中的内容，在程序框图中普通的字符串控件没有任何区别。

路径控件可以通过单击控件旁边自带的小按钮来打开文件选择对话框，确定文件路径。通过路径字符串之间的转换函数就能很容易地实现两者的转换，从而利用丰富的字符串函数对路径进行操作。

### 7.3.2 表格和树形控件

表格 (table) 和树形控件在控件选板中的位置如图 7.25 所示。对应的“系统 (System)”和

“经典（Classic）”下也能找到相应控件，使用方法相同，只是风格不一样。

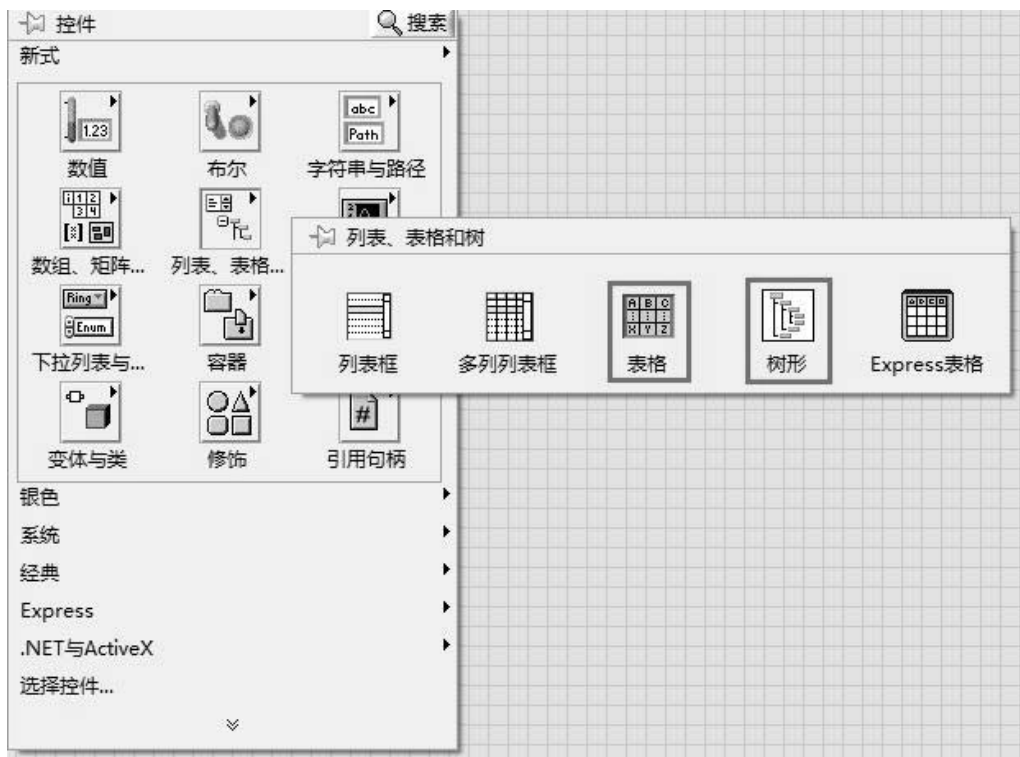


图 7.25 表格和树形控件在控件选板中的位置

表格实际上就是一个字符串组成的二维数组。将该控件放置在前面板后，可以直接右击该控件编辑它的各种属性。右击该控件，选择“显示项→行首（Visible Items→Row Headers）”或者“列首（Column Headers）”选项可以显示行头或列头。行头和列头作为说明性文字并不作为表格的实际内容。表格的编辑页非常简单，用鼠标单击对应的空格就能直接编辑内容了。右击该控件，选择“数据操作（Data Operations）”选项可以直接插入或删除整行或整列。表格数组的大小由输入内容所占的范围决定。

树形控件以树的形式显示多层内容，Windows 的资源管理器就是用树形控件来显示文件目录的。默认放置控件在前面板上时该控件有多列的输入，一般来说只有第一列有用，后面的列只是起到文字说明的作用。直接在需要输入内容的地方单击鼠标后就可以输入数据了，更多的操作只需要该控件的右击快捷菜单就能实现。

### 7.3.3 字符串函数

字符串相关的函数在函数选板中的位置如图 7.26 所示。这些 VI 函数基本涵盖了字符串处理所需要的各种功能。



图 7.26 字符串函数在函数选板中的位置

### 1) 日期时间\字符串转换函数 (Format Data/Time String)

该函数将“时间标识常量 (Time stamp)”类型的时间日期数据按设定的格式转换为字符串。日期时间\字符串转换函数如图 7.27 所示。

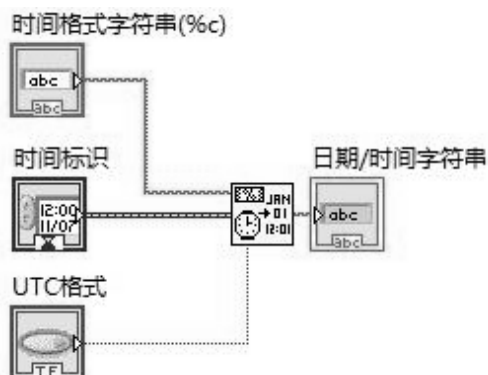


图 7.27 日期时间/字符串转换函数

具体的时间显示格式有：%a (显示星期几)、%b (显示月份)、%c (本地机器设定的日期时间格式)、%d (该月的第几天)、%H (小时——24 小时制)、%I (小时——12 小时

制)、%m (月)、%M (分钟)、%p (a.m, p.m 标志)、%S (秒)、%x (本地机器格式的日期)、%X (本地机器格式的时间)、%y (本世纪的年份)、%Y (包括本世纪的年份显示)、%<digit>u (按一定的精度显示秒的小数部分)。其中 UTC format 输入若为 True, 输出字符串将以国际标准时间为准。默认为 False, 以本机设定的时间区域为准。

## 2) 数值\字符串转换函数

在 LabVIEW 程序中经常需要将采集到的数据转换为字符串形式存储在文件中, 或从文件中读取数据形式的字符串转换为数据, 因此需要经常用到数值\字符串转换函数。

最简单的单个数值\字符串转换函数有“数值至十进制数字字符串转换 (Number to Decimal String)”、“数值至十六进制数字字符串转换 (Number to Hexadecimal String)”、“数值至工程字符串转换 (Number To Engineering String)”、“数值至指数数字字符串转换 (Number To Exponential String)”和“数值至小数数字字符串转换 (Number To Fractional String)”。分别对应将输入数据转换为十进制形式、十六进制形式、工程数据形式、指数形式和小数形式的字符串。当然也有一一对应的反转函数将这些字符串重新转换为数据。

## 3) 数组至电子表格字符串转换函数

数组至电子表格字符串转换函数如图 7.28 所示。

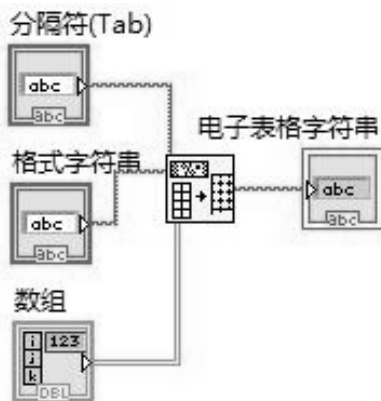


图 7.28 数组至电子表格字符串转换函数

直接将数据数组转换为字符串数组可以通过 Express Table 实现。若需要将数据数组存入 Excel 之类的电子表格中, 则需要将不同单元格中的字符串用 Tab 键隔开。

数组至电子表格字符串转换函数不仅能转换一维和二维数组, 还能转换三维或更多维的数组, 但是会自动在字符串对应的地方加入一些附加头信息。因此, 最好的存储方法是将多维数组转换为二维数组存储, 读取时再将其重新转换为多维数组。

**例 7-2** 输入两个字符串, 将它们拼成一个字符串后输出; 按需要截取部分字符串并实现字符串的替换。

(1) 启动 LabVIEW 2013, 打开前面板, 选择控件选板中的“新式→字符串与路径→字符串输入控件”, 将两个字符串输入控件置于前面板上, 并命名为“字符串 a”和“字符串 b”。

(2) 切换到程序框图，选择函数“编程→字符串→连接字符串”函数，将其置于程序框图中，并将其输入端口与“字符串 a”和“字符串 b”的输出端口连接，在连接字符串函数上单击鼠标右键，在弹出的快捷菜单中选择“创建→显示控件”命令，创建“连接字符串”显示控件。

(3) 选择“编程→字符串→字符串长度”函数，将其置于程序框图中，在字符串长度函数上单击鼠标右键，在弹出的快捷菜单中选择“创建→显示控件”命令，创建一个“长度”显示控件并连线，如图 7.29 所示。

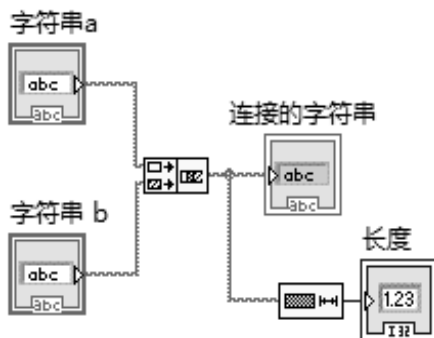


图 7.29 连接字符串并求其长度

(4) 选择“编程→字符串→截取字符串”函数，将其置于程序框图中，并创建两个输入控件和一个显示控件，分别命名为“偏移量”、“长度（剩余）”和“子字符串”并连线，如图 7.30 所示。

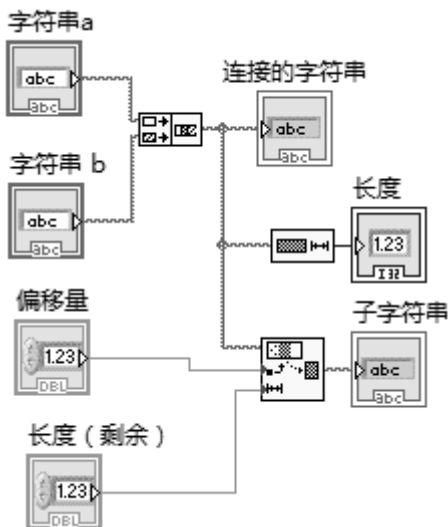


图 7.30 截取字符串

(5) 选择“编程→字符串→替换子字符串”函数，将其置于程序框图中，并创建三个输入控件和一个显示控件，分别命名为“替换的字符串”、“长度（子字符串长度）”、“偏移量”和“结果字符串”并连线，如图 7.31 所示。

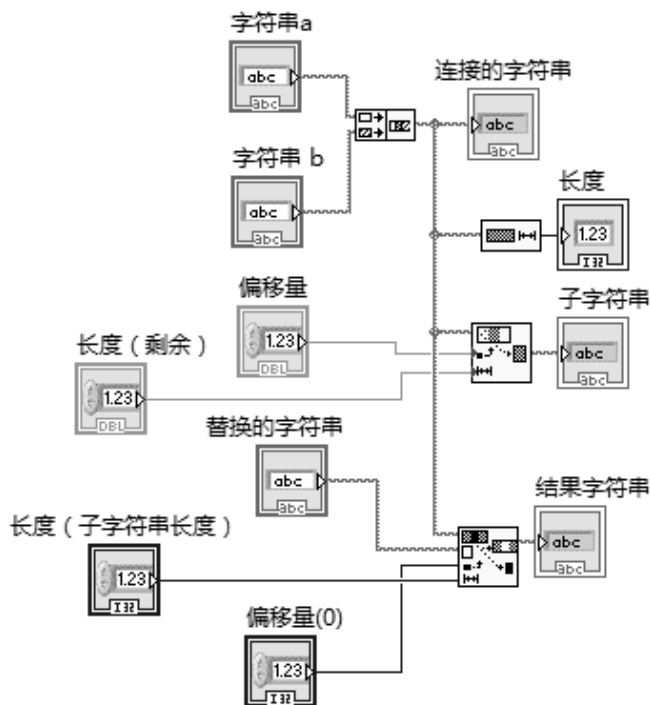


图 7.31 替换字符串

(6) 设置输入控件，运行程序，执行结果如图 7.32 所示。



图 7.32 程序执行结果

## 7.4 矩 阵

矩阵（Matrix）控件在控件选板中的位置如图 7.33 所示。有两种矩阵：一种为实数型矩阵，另一种为复数型矩阵。

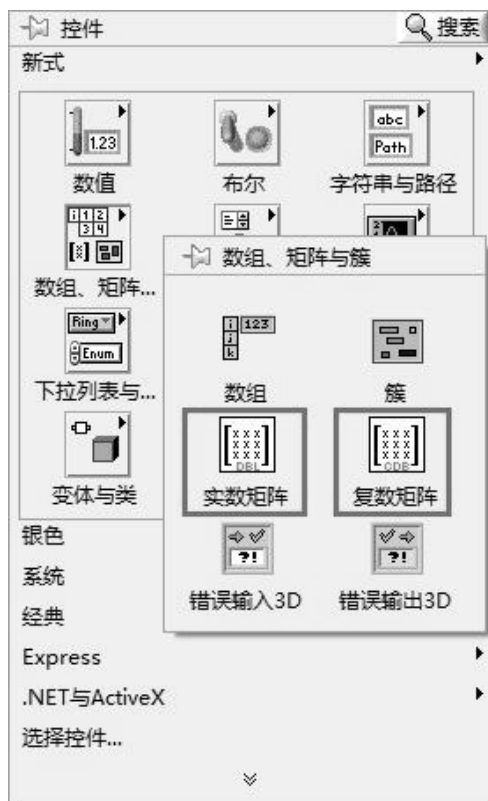


图 7.33 矩阵控件在控件选板中的位置

矩阵的运算方法和数组的运算方法有很大的不同，比如两个数组相乘是直接将相同索引的数组元素相乘，而矩阵的相乘必须按照线性代数中规定的方法相乘，因此用数组实现矩阵运算是非常麻烦的，而 LabVIEW 对矩阵的支持使矩阵的运算变得非常简单。矩阵控件分两种：一种为实数型矩阵，另一种为复数型矩阵。

如果直接将两个矩阵相乘，LabVIEW 会自动按照矩阵乘法相乘，输出也必然是矩阵。如果两个矩阵不满足乘法要求，则输出为空矩阵。

虽然矩阵和数组的运算是不同的，但是矩阵可以转换为二维数组，从而利用数组的函数对矩阵进行操作，操作完成后还可以再用转换函数将其转换为矩阵。

# 第 8 章 图形化显示

## 8.1 波形图

波形图（Waveform Graph）组件为事后记录波形控件，可以显示一个或者多个波形。波形图在控件选板中的位置如图 8.1 所示。

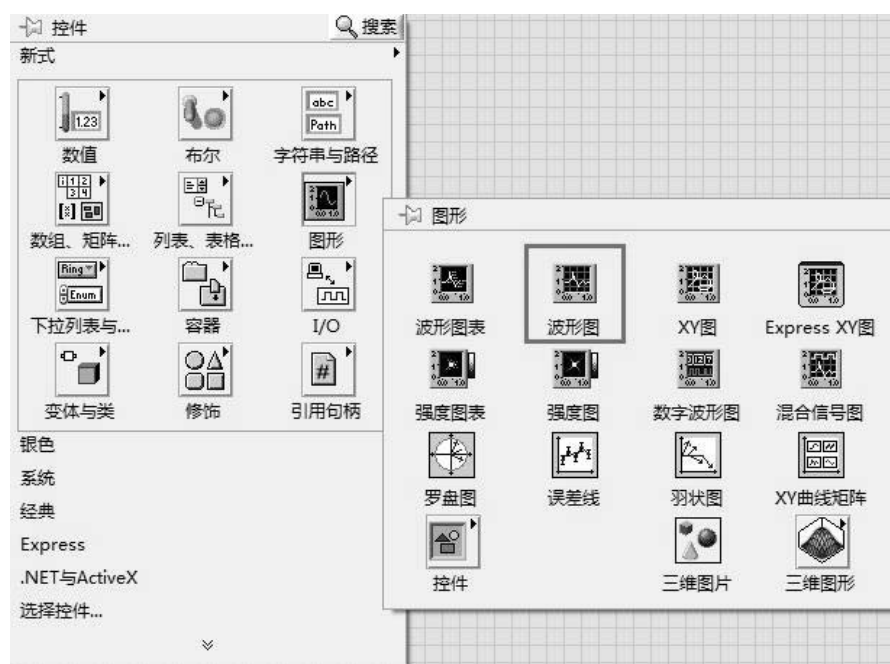


图 8.1 波形图在控件选板中的位置

波形图可以有多种数据输入类型：一维数组、二维数组、簇、簇数组、波形数据。

### 1. 一维数组作为输入

当输入数据为一维数组时，波形图直接将一维数组画成一条曲线，纵坐标为数组元素的值，横坐标为数组索引，如图 8.2 所示。

### 2. 二维数组作为输入

当输入数组为二维数组时，默认情况下每一行的数据对应一条曲线，即曲线的数目和行数相同，如图 8.3 所示。若右击图标并选择转置数组（Transpose Array）选项，则可以实现每一列的数据对应一条曲线，即曲线的数目和列数相同。



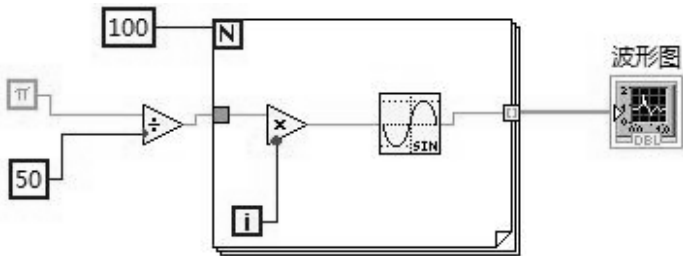
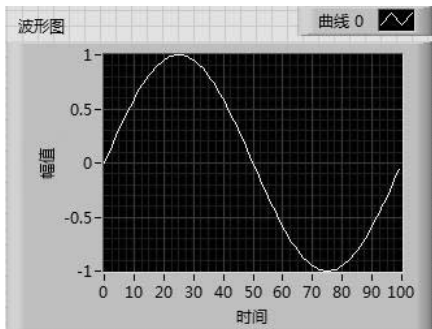


图 8.2 一维数组波形图

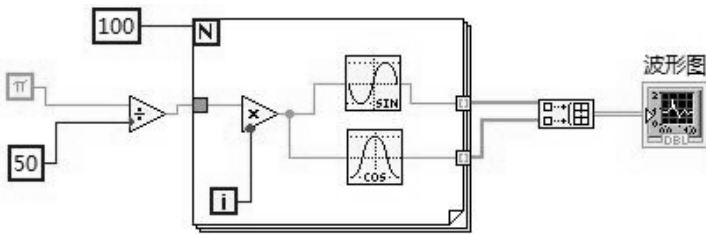
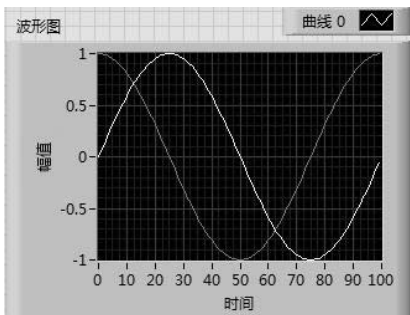


图 8.3 二维数组波形图

### 3. 簇作为输入

簇作为输入时需要指定 3 个元素：起始位置  $x_0$ 、数据点间隔  $dx$  和数组数据。由于指定了  $x_0$  和  $dx$ ，因此图表的横坐标不再是数组的索引，而是由  $x_0$  和  $dx$  决定。数组数据可以是一维数组、二维数组或一维簇数组。一维数组和二维数组作为簇的数组元素输入非常简单，如图 8.4 所示。

若两条曲线的点数不一样，则需要采用一维簇数组作为输入。首先需要将数组通过捆绑函数绑定为簇，再将簇组成一维数组作为数组数据。由于两条曲线的数组都绑定为簇，因此两条曲线的点数即使不相同也能在同一幅图中显示，如图 8.5 所示。其中曲线 1 点数为 100，曲线 2 点数为 200，但起始点和点之间的间隔都是一致的。

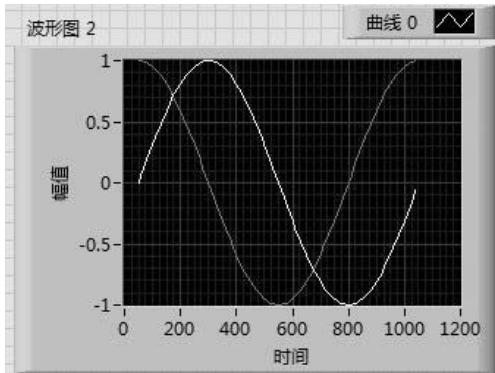
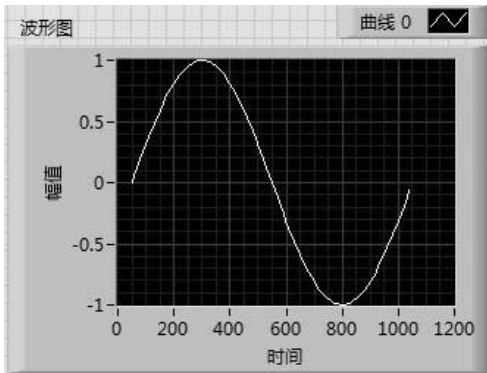


图 8.4 一维数组和二维数组作为簇输入的波形图

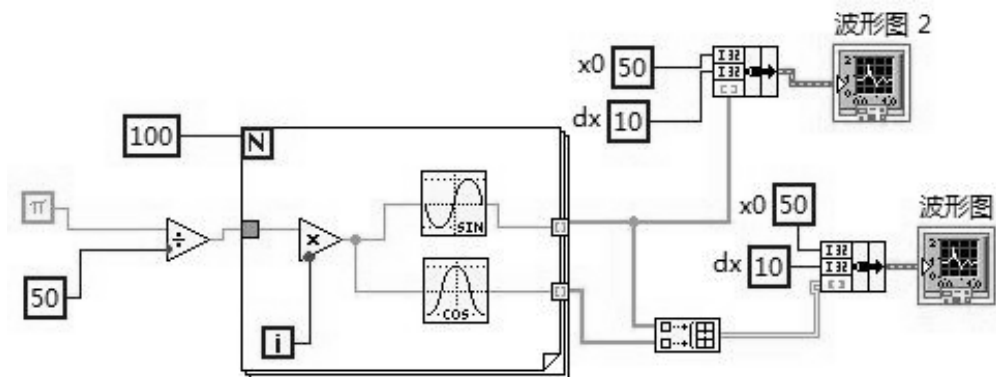


图 8.4 一维数组和二维数组作为簇输入的波形图 (续)

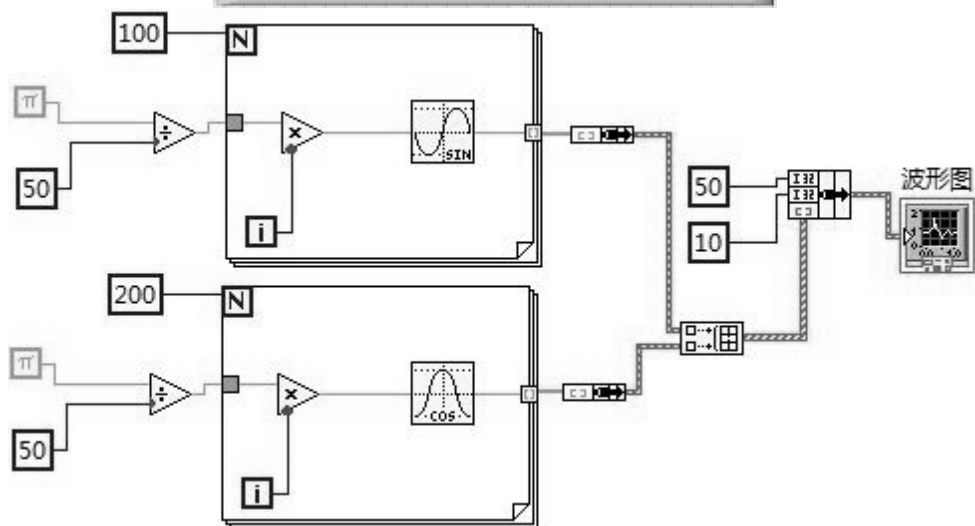
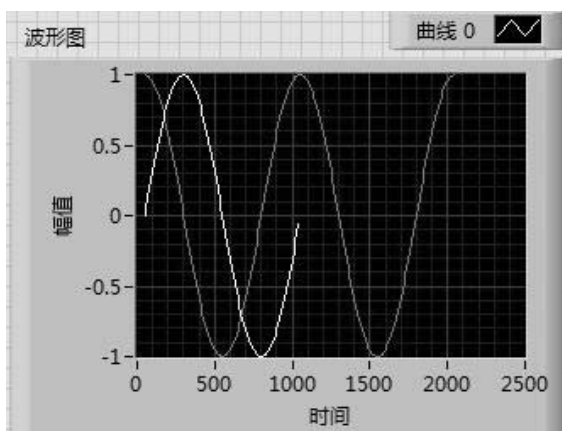


图 8.5 一维簇数组作为簇输入的波形图

#### 4. 簇数组作为输入

图 8.5 中所示的一维簇数组也可以直接作为图表的输入, 此时相当于  $x_0$  为 0,  $dx$  为 1, 如图 8.6 所示。

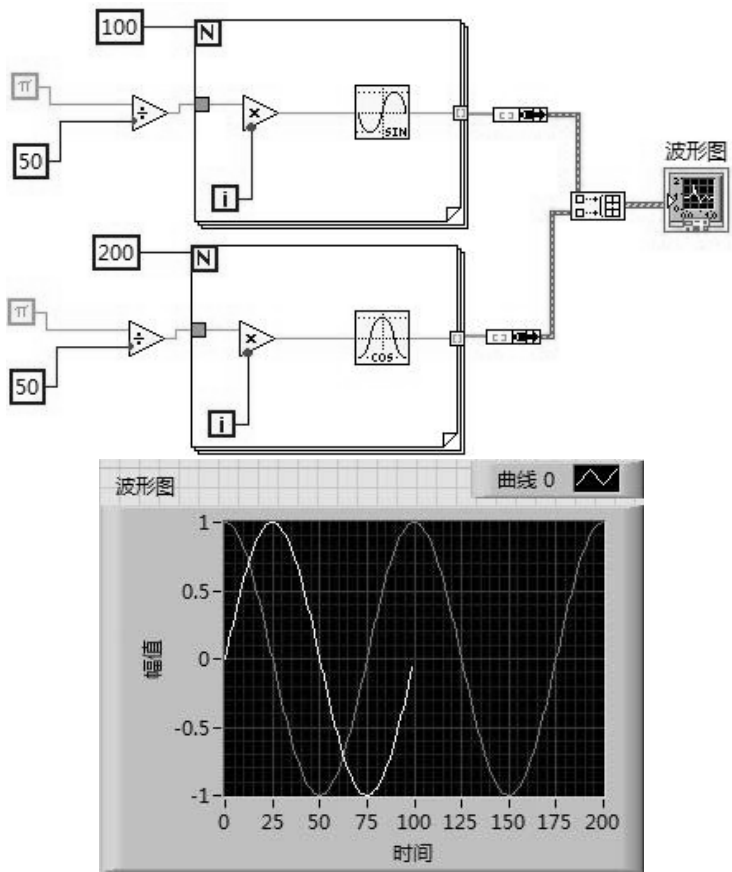


图 8.6 一维簇数组直接作为输入的波形图

5. 波形数据作为输入

由于波形数据所携带的数据横轴为时间，因此需要将波形图的横轴设为时间轴，如图 8.7 所示。若把多个波形数据构造在一维数组中，则能显示多条曲线，类似于簇数组作为输入。

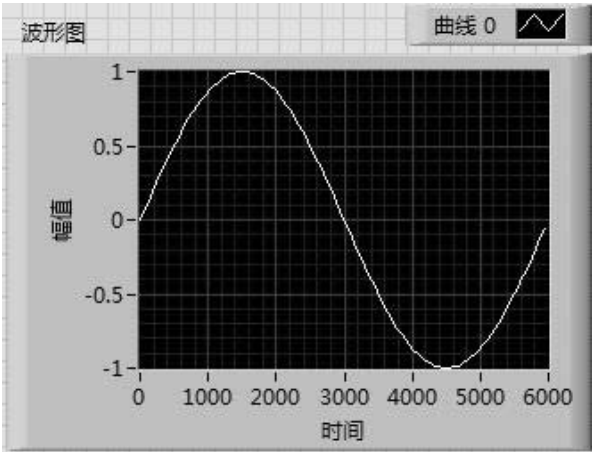


图 8.7 波形数据作为输入的波形图

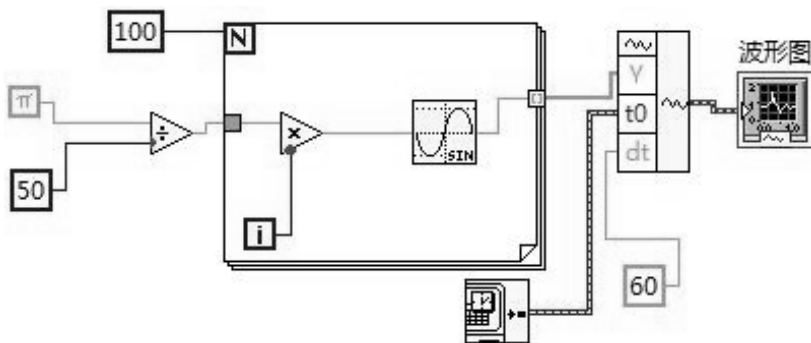


图 8.7 波形数据作为输入的波形图（续）

波形图与波形图表的前面板是有所不同的。波形图表的  $X$  轴只在起始和结束的位置才有刻度，而波形图的  $X$  轴根据实际情况的最大值要求，刻度均匀地分布。

此外，由于两者的波形刷新方式和数据组织方法的不同，波形图表的输入是一个双精度浮点变量，波形图的输入则为双精度浮点数组。

对于波形图表控件来说，它把新的数据连续扩展到已有数据的后面，波形是连续向前推进显示的，这种显示方法使用户可以很清楚地观察数据的变化过程：对于波形图控件来说，它通常把需显示的数据先收集到一个数组中，然后再把这组数据一次性在控件中显示。波形图控件适合用于实时测量中的参数监控，而波形图表控件适合用于事后数据的分析。

## 8.2 波形图表

波形图表（Waveform Chart）是显示一条或者多条曲线的特殊数值控件，一般用于显示以恒定速率采集得到的数据。波形图表会保留此前更新的历史数据，又称为缓冲区。右键单击图表，从快捷菜单中选择“图标历史长度”，可配置缓冲区大小。波形图表的默认图表历史长度为 1024 个数据点。向图表传送数据的频率决定了图表重绘的频率。

波形图表接收的数据类型包括标量数据、一维数组、波形数据和二维数组。通过簇绑定的方法可以显示多条曲线。

对于标量数据，波形图表直接将数据添加在曲线的尾端，如图 8.8 所示。

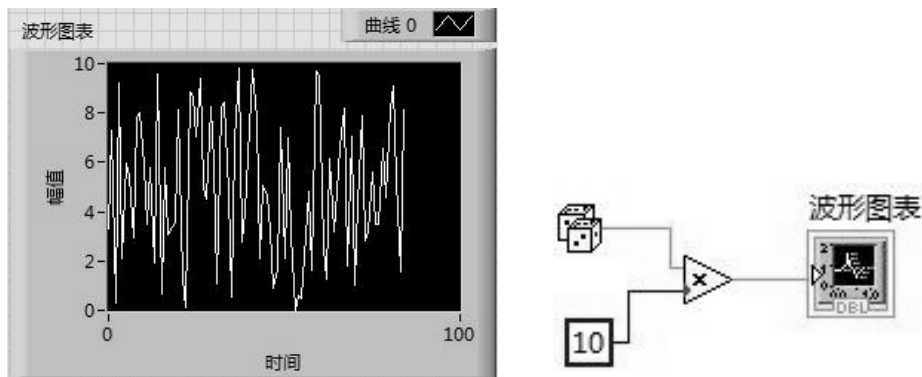


图 8.8 单曲线标量数据的波形图表

对于一维数组数据，它会一次性把一维数组的数据添加到曲线末端，即曲线每次向前推进的点数为一维数组的点数，如图 8.9 所示。

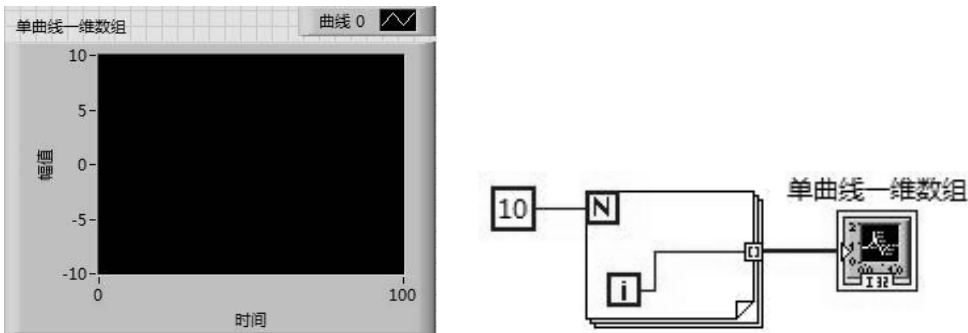


图 8.9 单曲线一维数组的波形图表

如果要显示多条标量曲线，只需要用簇的捆绑函数将它们绑定在一起作为输入即可，如图 8.10 所示。

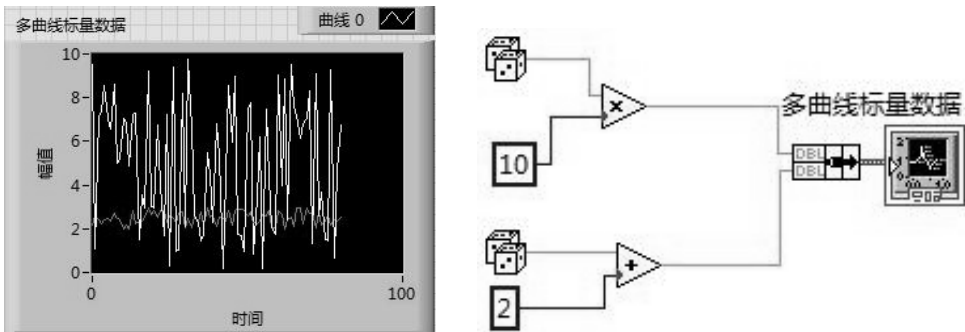


图 8.10 在同一波形图表中显示多条标量数据曲线

对于二维数组，默认情况下它将输入数组转置，即每一列的数据当作一条一维数组曲线，图 8.11 所示的 5 行 3 列的二维数组在图中显示为 3 条曲线，每递进一步每条曲线增加 5 个点。若右击图表取消选择“转置数组 (Transpose Array)”选项，则它把每一行的数据当作一条一维数组曲线。

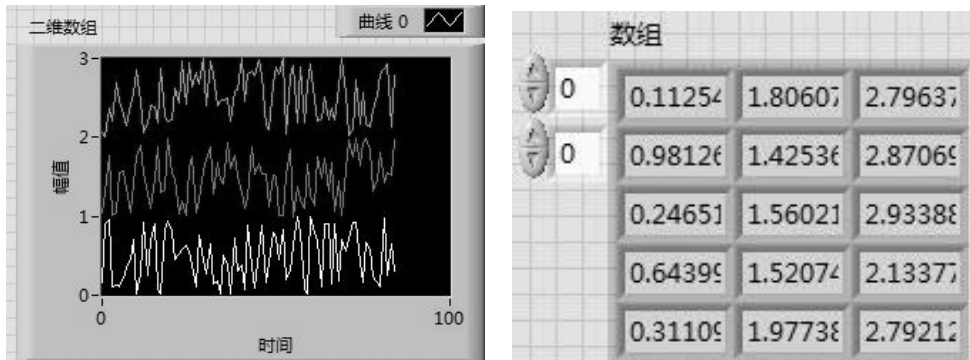


图 8.11 二维数组的波形图表

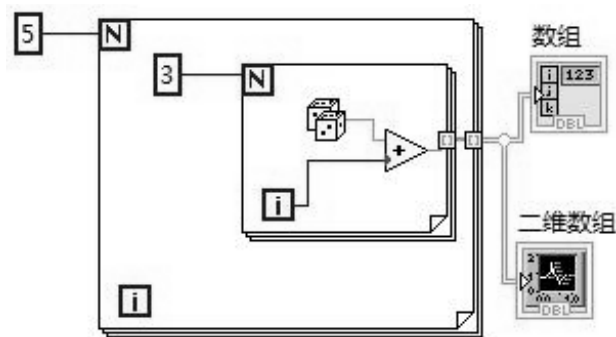


图 8.11 二维数组的波形图表（续）

对于波形数据类型，波形图表只能显示当前的输入数据，并不能将新数据添加到曲线尾端。这是因为波形图表包含了横坐标的数据，因此每次画出的结果和上一次结果都无关，此时波形图表等价于波形图。

### 8.3 XY 图

当用户需要画的曲线是由  $(x, y)$  坐标决定的时候，就可以采用 XY 曲线图。其实波形图在一定意义上也是 XY 曲线图，但是它的  $X$  轴必然是等间距的，而且不可控制。与之相比，XY 曲线图就要灵活多了，它的  $X$  轴坐标由输入的  $X$  轴决定， $Y$  轴坐标由  $Y$  轴数组决定，因此它还能画各种封闭曲线。它在控件选板中的位置如图 8.12 所示。

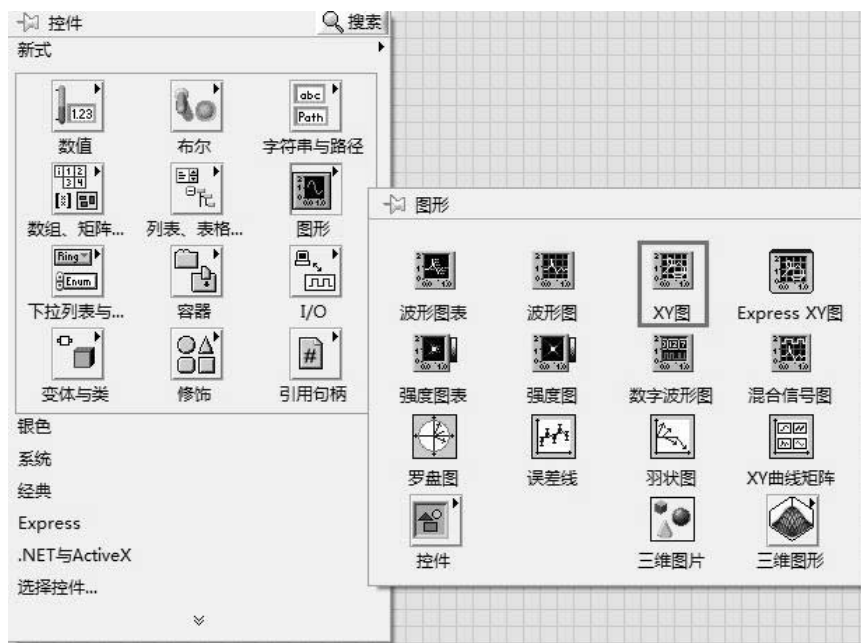


图 8.12 XY 图在控件选板中的位置

XY 曲线图的输入数据类型相对比较简单，一种是直接将  $X$  数组和  $Y$  数组绑定为簇作为输入，另一种是把每个点的坐标都绑定为簇，然后作为簇数组输入，对于两种方式，都可以通过将多个输入合并为一个一维数组输入来实现一幅图中显示多条曲线。

### 1. XY 两个一维数组绑定为簇作为输入

这是最简单的一种情形，捆绑函数输入第一个数组为  $X$  数组，第二个数组为  $Y$  数组。绑定为簇后可以直接输入，也可以将多个簇构造为一维数组输入，实现多条曲线，如图 8.13 所示的李萨如图形和一个圆。

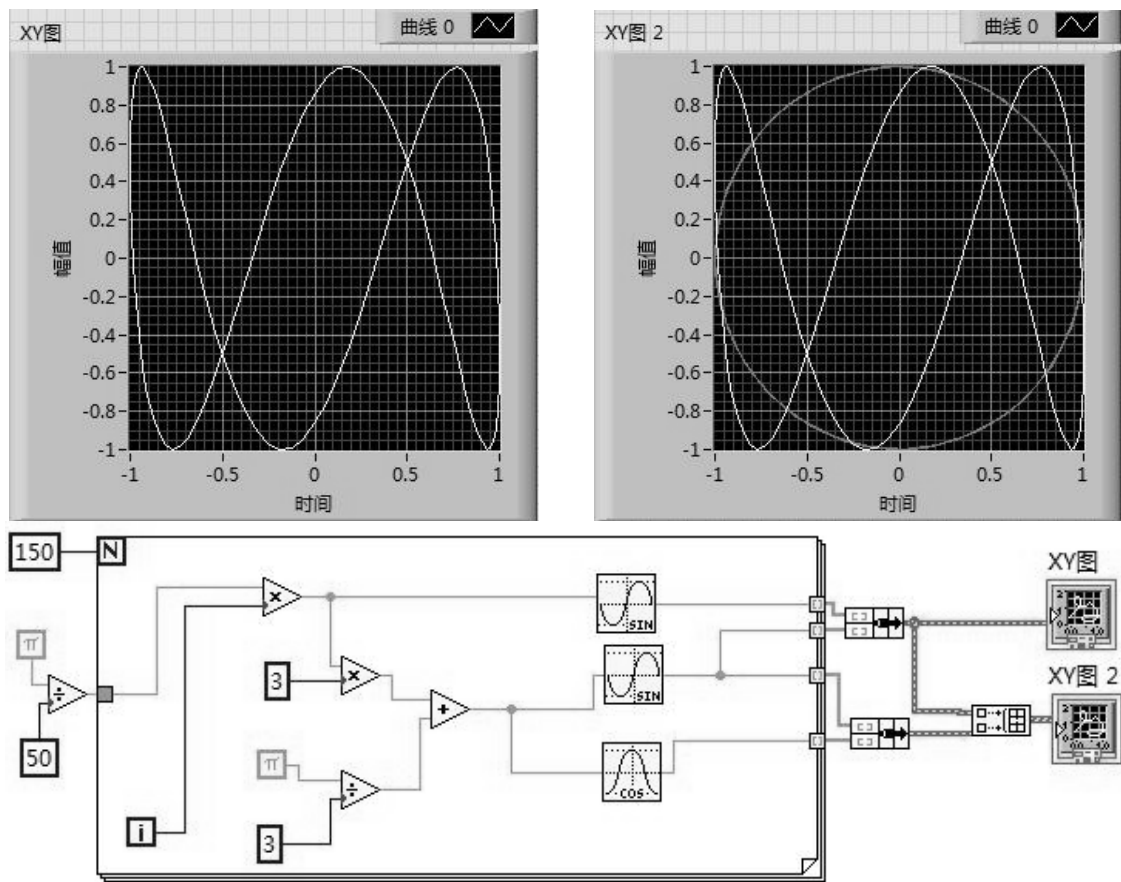


图 8.13 一维数组簇作为输入的 XY 曲线图示例

### 2. 坐标点簇数组作为输入

该方式将各个点的坐标绑定为簇然后作为簇数组输入，和直接将 XY 数组绑定为簇输入效果一样。但是后者不一定需要 XY 数组的大小一致，它会自动将大的数组裁剪，至于采用哪种方法则根据个人爱好和编程情况而定。若要将多个坐标点簇数组作为输入，则首先需要将它绑定为簇，然后构造为一维数组作为输入，从而实现绘制多条曲线。将图 8.13 的程序改写为该方式的输入后如图 8.14 所示。

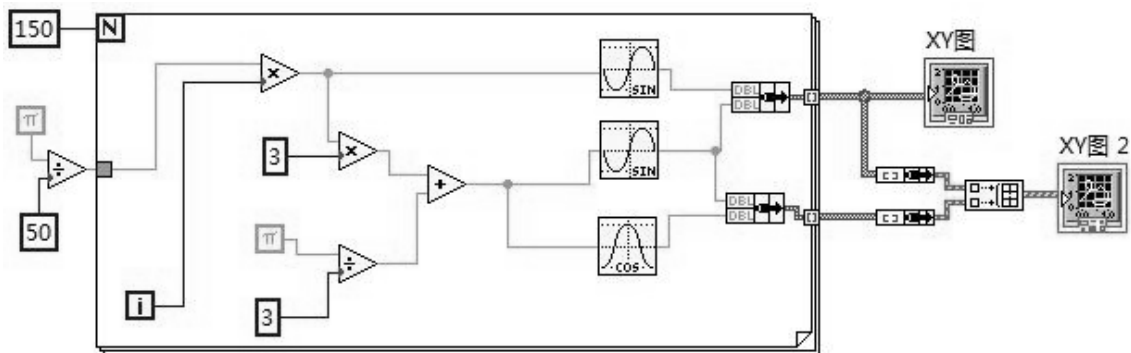


图 8.14 坐标点簇数组作为输入的 XY 曲线图示例

### 3. 时间作为 X 轴——利用 XY 曲线图实现历史曲线

在很多情况下，需要分析采集数据与时间的关系，即历史曲线。利用 XY 曲线图实现历史曲线非常简单。图 8.15 所示的是绘制历史曲线的程序框图，其中时间的间隔设置为随机，曲线记录了数据和采集时间的关系。

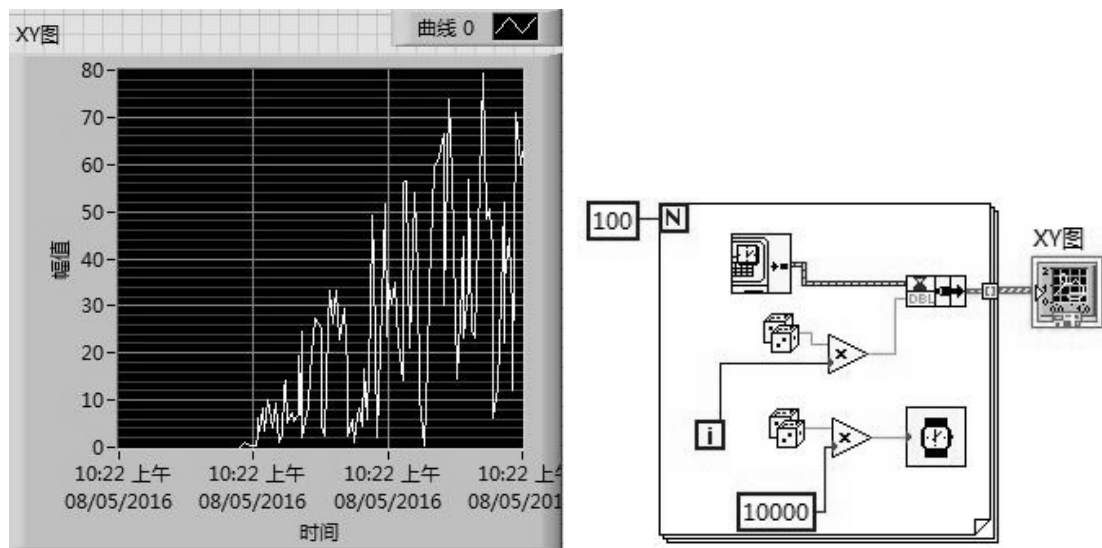


图 8.15 绘制历史曲线的程序框图

### 4. Express XY 曲线图

Express XY 曲线图在控件选板中的位置如图 8.16 所示。

Express XY 曲线图采用了 LabVIEW 的 Express 技术，将 Express XY 曲线图放置在前面板上的同时，在程序框图中会自动添加一个 VI，它的 XY 轴数据为动态数据类型。因此只需要将 XY 数组数据与之连接，就会自动添加一个转换函数，将其转换为动态数据类型。双击该函数可以选择是否在画新图时先清空画面。因此 Express XY 曲线图使用起来非常方便，如图 8.17 所示。



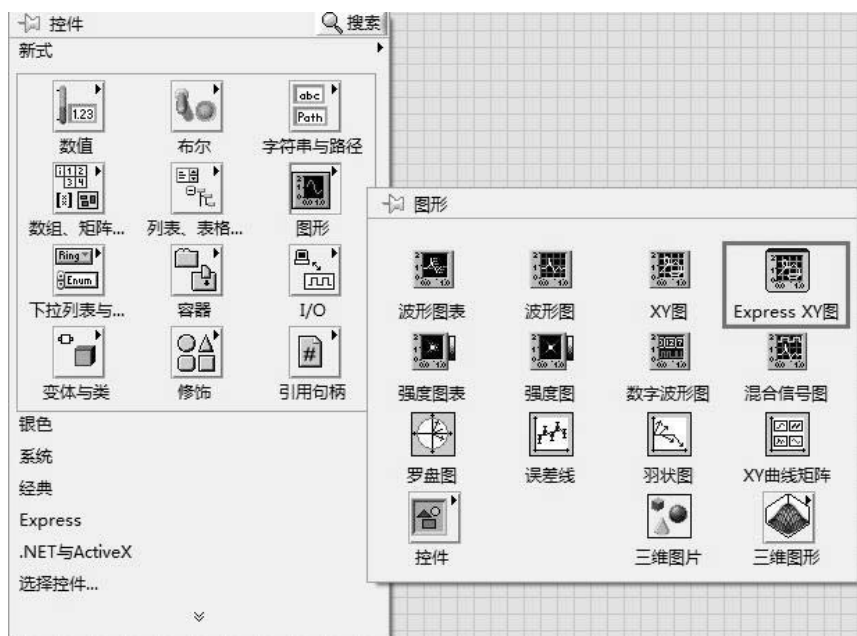


图 8.16 Express XY 曲线图在控件选板中的位置

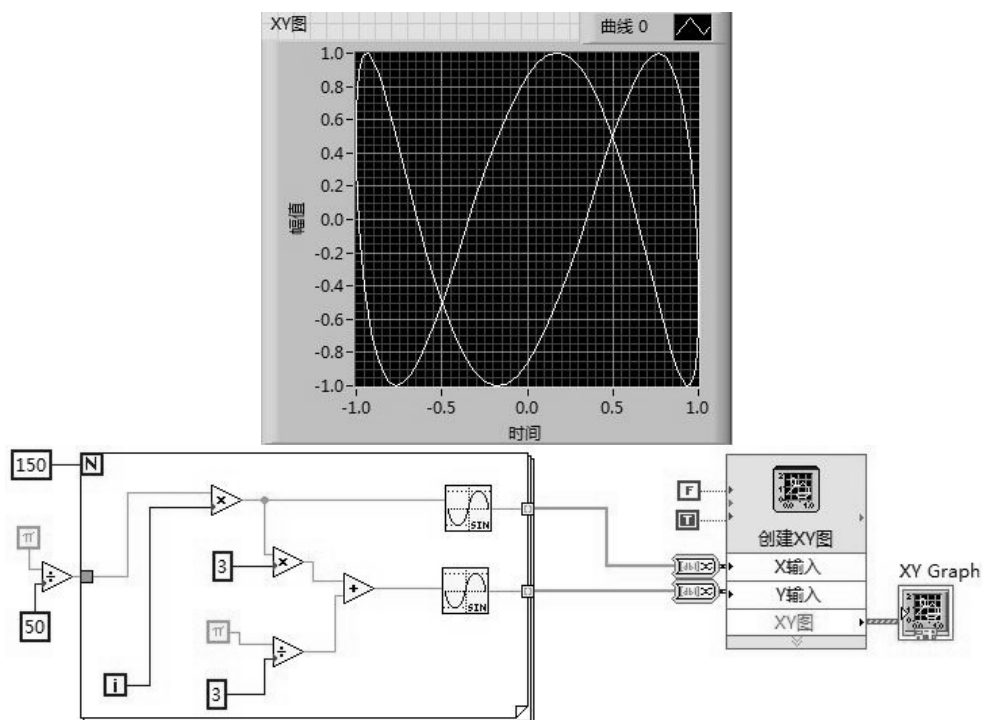


图 8.17 Express XY 曲线图示例

例 8-1 计一个 VI 来绘制李萨如图形。

(1) 启动 LabVIEW 2013, 打开程序框图, 选择函数选板的“信号处理→波形生成→正弦波”函数, 将两个正弦波函数置于程序框图中。

(2) 设置两个正弦波形 VI 的输入参数。第一个正弦波形的输入参数均使用默认值，第二个正弦波形的初始相位由第一个输入控件给定。在其相位端口上单击鼠标右键，在弹出的快捷菜单中选择“创建→输入控件”命令，在前面板创建一个输入值输入控件来输入第二个正弦波的初始相位。通过改变第 2 个正弦波形的初始相位来改变两个正弦波形的相位差。

(3) 选择“编程→波形→获取波形成分”函数，放置两个获取波形成分函数来分别获取两个正弦波形的 Y 分量。

(4) 选择“编程→簇、类与变体→捆绑”函数，将两个 Y 数组通过一个捆绑函数捆在一起，连接在 XY 图上。完成的程序框图如图 8.18 所示。

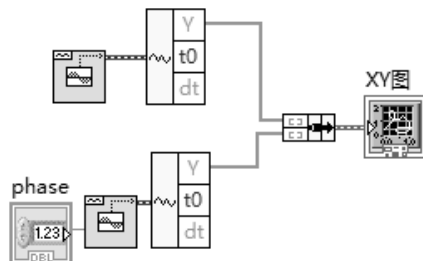


图 8.18 程序框图

(5) 切换到前面板，在相位控件中输入相位  $0^\circ$ 、 $45^\circ$  和  $90^\circ$ ，分别运行 VI，观察图形，结果如图 8.19、图 8.20 和图 8.21 所示。

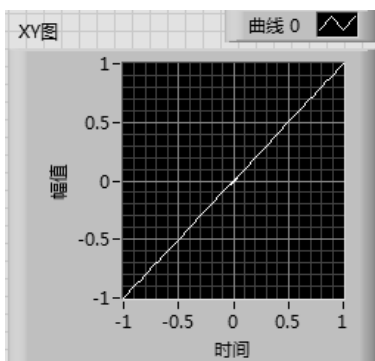


图 8.19 相位差为  $0^\circ$  时的李萨如图形

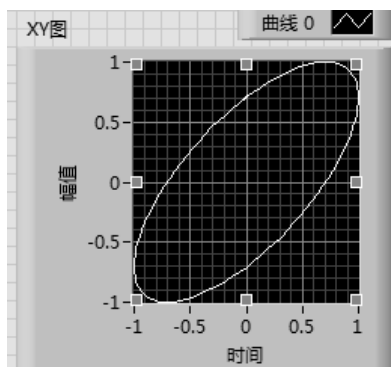


图 8.20 相位差为  $45^\circ$  时的李萨如图形

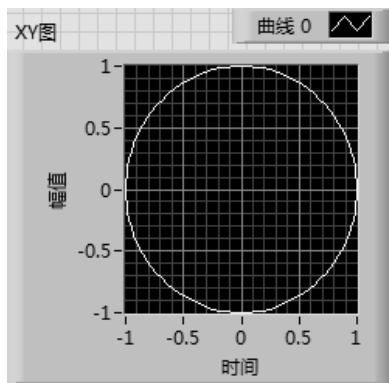


图 8.21 相位差为  $90^\circ$  时的李萨如图形

# 第9章 文件操作

文件操作与管理是测试系统软件开发的重要组成部分，数据存储、参数输入和系统管理都离不开文件的建立、操作和维护。LabVIEW 为文件的操作与管理提供了一组高效的 VI 集。文件 I/O 在函数选板中的位置如图 9.1 所示。



图 9.1 文件 I/O 在函数选板中的位置

## 9.1 文件操作的 VI 和函数

LabVIEW 的文件 I/O 操作是通过其 I/O 节点来实现的，这些 VI 和函数节点位于 I/O 子

选板中。本节将对这些 VI 和函数及其使用方法进行介绍。

## 1. 用于常用文件 I/O 操作的 VI 和函数

函数选板中文件 I/O 选板上的 VI 和函数可用于常见文件 I/O 操作，如读写以下类型的数据：在电子表格文本文件中读写数值，在文本文件中读写字符，从文本文件读取行，在二进制文件中读写数据。

可将读取文本文件、写入文本文件函数配置为可执行常用文件 I/O 操作。这些执行常用操作的 VI 和函数可打开文件或弹出提示对话框，要求用户打开文件，执行读写操作后关闭文件，节省了编程时间。如果“文件 I/O”VI 和函数被设置为执行多项操作，则每次运行时都将打开关闭文件，所以尽量不要将它们放在循环中。执行多项操作时可将函数设置为始终保持文件打开。文件 I/O 选板如图 9.2 所示。



图 9.2 文件 I/O 选板

下面对文件 I/O 选板中的节点进行介绍。

## 2. 写入电子表格文件 (Write To Spreadsheet File.vi)

写入电子表格文件的连线板如图 9.3 所示。

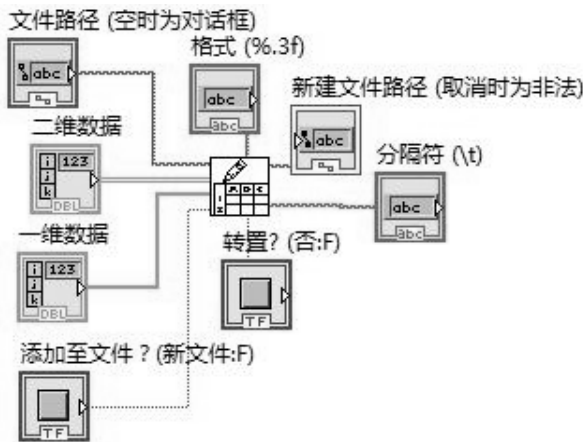


图 9.3 写入电子表格文件的连线板

将字符串型、浮点型或者整型的二维数组或者一维数组转换成电子表格，并写入一个新文件中，或追加到已有文件的末尾。

可输入参数如下。

- 格式：指定将数据转换成字符串的格式。
- 文件路径：待写入文件的路径，如果未连接则弹出对话框由用户选择。
- 二维数据和一维数据：待写入的数据数组，根据数组维数选择其中一个连接即可。
- 添加至文件？：指定是否追加到文件末尾，默认为假，代表开启新文件。
- 转置？：指定是否将数据数组转置，默认为假。
- 分隔符：每一条数据中用于分隔数据域的分隔符，默认为制表符。

输出参数“新建文件路径”为写入的文件路径。

### 3. 读取电子表格文件（Read From Spreadsheet.vi）

读取电子表格文件的连线板如图 9.4 所示。

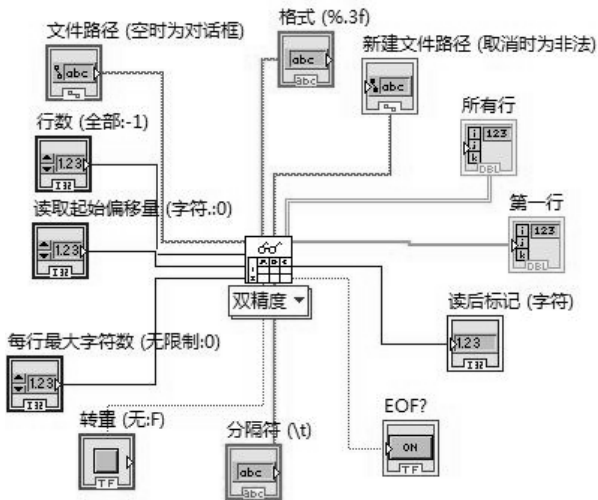


图 9.4 读取电子表格文件的连线板

从存储了电子表格的文件中读取数并转换成字符串型、浮点数值型或者整型的二维数组或者一维数组。

可输入参数如下。

- 格式：指定将数据转换的格式。
- 文件路径：待读取文件的路径，如果未连接则弹出对话框由用户选择。
- 行数：指定读取数据行数，默认为-1，表示读取所有数据。
- 读取起始偏移量：指定从文件中开始读取数据的位置，以字符数为单位。
- 每行最大字符数：在读取到文件末尾前最多读取字符数，默认为 0，表示无限制。
- 转置：文件中的数据是否以转置形式存入，默认为假。
- 分隔符：每一条数据中用于分隔数据域的分隔符，默认为制表符。

输出参数包括：

- 新建文件路径：文件路径。
- 所有行：从文件中读取出的所有数据，为二维数组。
- 第一行：“所有行”的第一行，为一维数组，一般用于从文件中直接读出一维数组。
- 读后标记（字符）：本次读取后的文件标识符位置，即最后一个被读取字符的下一字符位置，以字符数为单位，一般用于指定下次读取时的起始位置。
- EOF?：表示是否已经读取到文件尾。

#### 例 9-1 读写电子表格文件。

(1) 启动 LabVIEW 2013，打开程序框图，选择函数选板的“编程→结构→For 循环”函数，右击 For 循环体的“循环总数 N”创建常量，设置循环次数为 100；选择“编程→初等与特殊函数→三角函数→正弦”、“编程→数学→初等与特殊函数→三角函数→正弦”、“编程→数学→初等与特殊函数→三角函数→余弦”函数和“编程→数值→除”，置于 For 循环体内，利用 For 循环产生 100 个正弦数据和余弦数据，连线如图 9.5 所示。

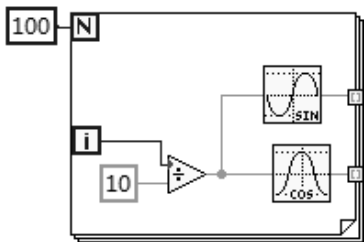


图 9.5 产生 100 个正弦数据和余弦数据

(2) 选择“编程→数值→创建数组”和“编程→文件 I/O→写入电子表格文件”函数，将其置于 For 循环体外，向下拖动“创建数组”边框，增加一个“元素”接口，并将其与“正弦”和“余弦”函数的输出口连接，“创建数组”的输出端与“写入电子表格文件”的输入端连接，在“写入电子表格文件”的“文件路径”上右击创建输入控件，并在前面板中通过该控件选择文件保存路径。写入电子表格文件的程序框图如图 9.6 所示。

(3) 运行文件，会自动在保存路径上创建一个电子表格文件。

(4) 选择“编程→文件 I/O→读取电子表格文件”函数，将其路径与“写入电子表格文件”的“文件路径”相连。

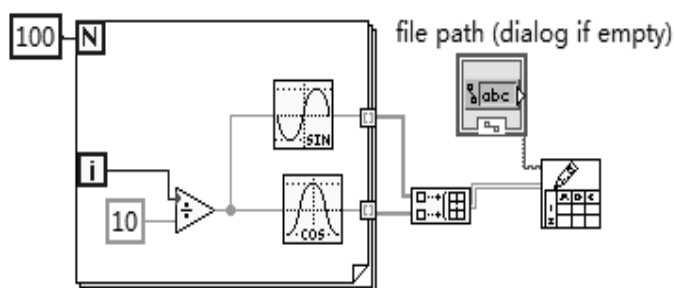


图 9.6 写入电子表格文件的程序框图

(5) 切换到前面板中, 选择控件选板中的“新式→图形→波形图”控件, 在程序框图中将“读取电子表格文件”的输出端与之相连。保证写入数据后再读取, 添加一个顺序结构, 如图 9.7 所示。

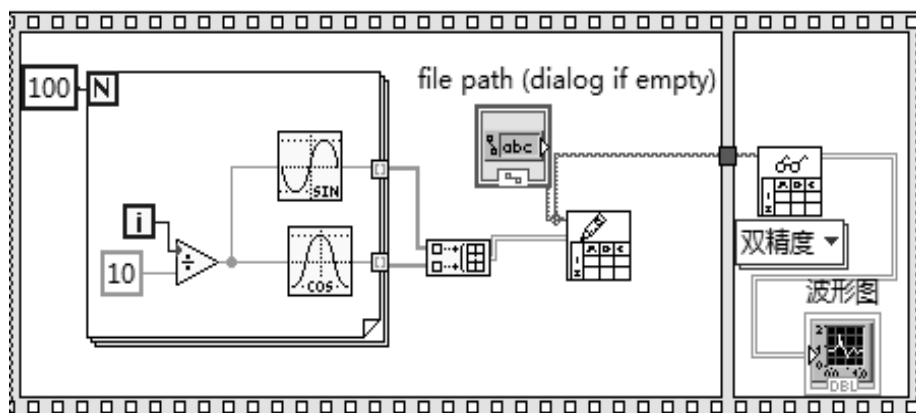


图 9.7 读写电子表格文件的程序

(6) 运行 VI, 可看到波形图上显示了从电子表格文件中读取的正弦数据和余弦数据, 如图 9.8 所示。

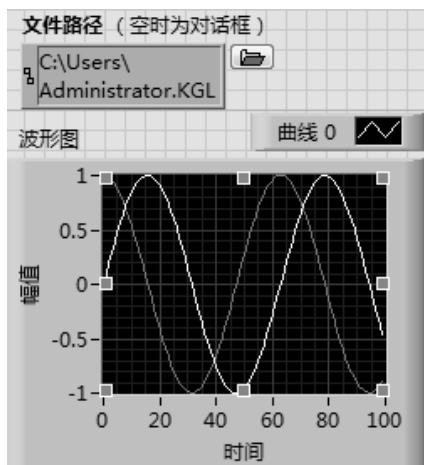


图 9.8 运行结果

#### 4. 打开/创建/替换文件（Open/Create/Replace Files）

打开/创建/替换文件的连线板如图 9.9 所示。

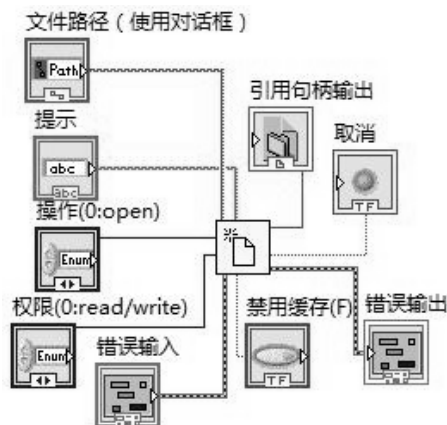


图 9.9 打开/创建/替换文件的连线板

用于打开、创建或替换文件，其可输入参数如下所述。

- 提示：在文件对话框中出现的提示信息文本。
- 文件路径：若未连接则弹出对话框由用户选择文件路径。
- 操作：指定操作功能，为枚举量。
- 权限：指定文件读写方向的枚举量，分为 0——读和写，1——只读，2——只写。
- 错误输入：输入错误信息簇。

输出参数包括以下内容。

- 引用句柄输入：为文件分配的引用句柄，可用于下一步对文件进行进一步操作，如读取、写入、关闭等。
- 取消：如果在文件路径选择对话框中选择了取消，或者在是否替换已有文件对话框中选择了否时，为真。
- 错误输出：输出错误信息簇。

#### 5. 关闭文件（Close File）

关闭文件的连线板如图 9.10 所示。



图 9.10 关闭文件的连线板

用于关闭已经打开的文件。通过引用句柄参数获得文件并关闭，并返回该文件的路径。



输入输出错误信息簇在该函数中的应用与一般情况有所不同,无论前面的操作是否有错误信息产生,本函数都将照常执行,以确保文件被关闭。

## 6. 格式化写入文件 (Format Into File) 与扫描文件 (Scan From File)

格式化写入文件与扫描文件的连线板,如图 9.11 所示。

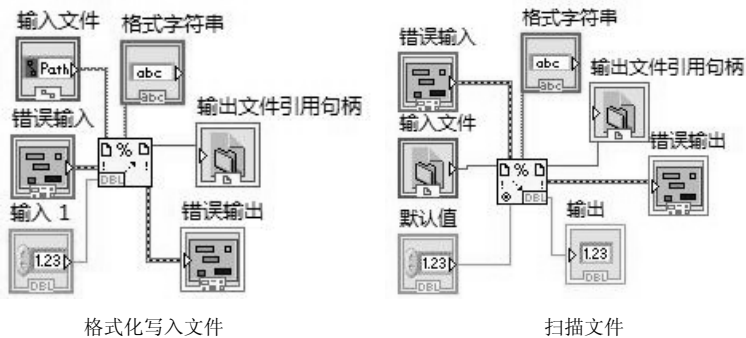


图 9.11 格式化写入文件与扫描文件的连线板

从文件中提取字符型、数值型、布尔型或路径型的数据,或者将这些类型的数据按照指定格式写入文件中。“格式化写入文件”和“扫描文件”函数的用法与字符串处理中的“格式化写入字符串”和“扫描字符串”函数的用法非常相似,不同之处仅在于操作对象不同,另外需要指定文件的引用句柄,即参数“输入文件”。

## 7. 写入文本文件 (Write To Text File) 与读取文本文件 (Read From Text File)

写入文本文件与读取文本文件的连线板如图 9.12 所示。

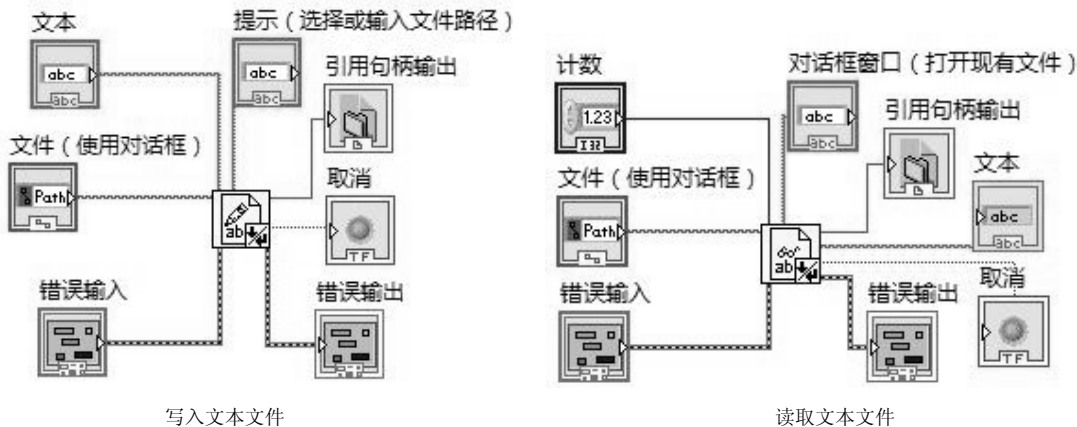


图 9.12 写入文本文件与读取文本文件的连线板

写入文本文件函数用于将文本写入文件中,读取文本文件函数则用于从文件中读取文本。两者共有的参数如下所述。

- 对话框窗口: 对话框窗口中将要出现的提示信息文本。
- 文件: 可以为文件路径或者引用句柄。如果为路径,函数将打开路径所指定的文件

然后写入，如果文件不存在则创建一个新文件。

- 文本：待写入或读取出的文本。
- 错误输入和错误输出：输入输出错误信息簇。
- 引用句柄输出：为文件分配的引用句柄，可用于下一步对文件进行进一步操作，如读取、写入、关闭等。
- 取消：在文件路径选择对话框中选择取消时为真。

除此之外，读取文本文件函数多一个输入参数“计数”，用来指定在遇到文件末尾时最多读取的字符数或行数，如果“计数”小于 0 则不限制。函数的右键菜单中“读取行”项用于选择读取方式为读取行或读取字符，若读取行则默认情况下仅读取一行，若读取字符则默认情况下为读取整个文件。

## 8. 写入二进制文件（Write To Binary File）和读取二进制文件（Read From Binary File）

写入二进制文件和读取二进制文件的连线板如图 9.13 所示。

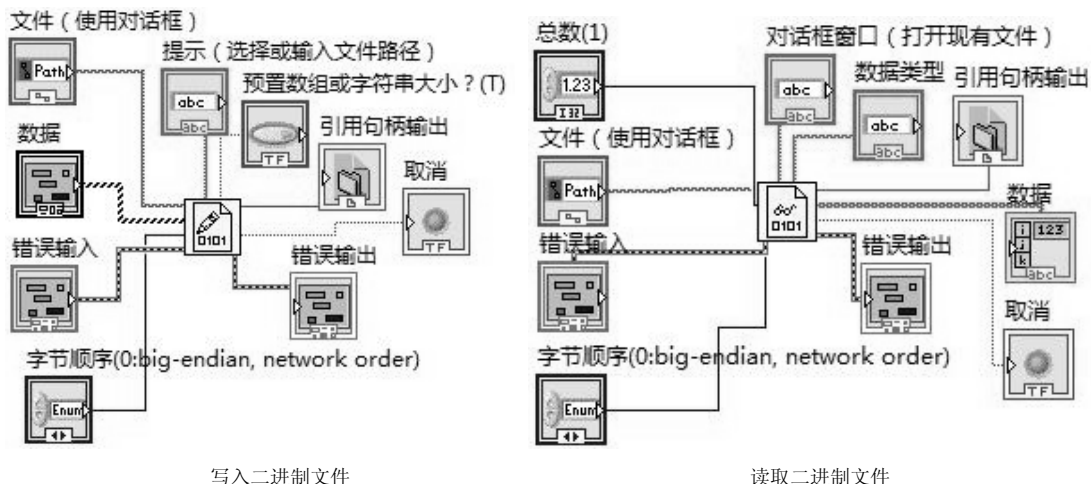


图 9.13 写入二进制文件和读取二进制文件的连线板

写入二进制文件函数用于将数据写入二进制文件中，读取二进制文件函数则用于从二进制文件中读取数据。两者共有的参数如下。

- 对话框窗口：对话框窗口中将要出现的提示信息文本。
- 文件：可以为文件路径或者引用句柄，如果为路径，函数将打开路径所指定的文件然后写入，如果文件不存在则创建一个新文件。
- 数据：待写入或读取出的数据，可以为任意类型的数据。
- 字节顺序：指定字节次序，为枚举型变量。
- 错误输入和错误输出：输入/输出错误信息簇；
- 引用句柄输出：为文件分配的引用句柄，可用于下一步对文件进行进一步操作，如读取、写入、关闭等。
- 取消：在文件路径选择对话框中选择取消时为真。

此外，写入二进制文件函数还拥有以下参数。

- 预置数组或字符串大小?：指定 LabVIEW 是否在输出参数引用句柄输出的头部包含数据大小信息（仅当数据类型为数组或字符串时）。

读取二进制文件函数还拥有以下参数。

- 数据类型：设定读取数据的类型。
- 总数：指定数据元素的被读取数量，以字节或者元素个数为单位，默认只读取一个数据元素，如果“总数”为-1，则读取所有数据元素。

## 9. 创建路径（Build Path）和拆分路径（Strip Path）

创建路径和拆分路径的连线板如图 9.14 所示。

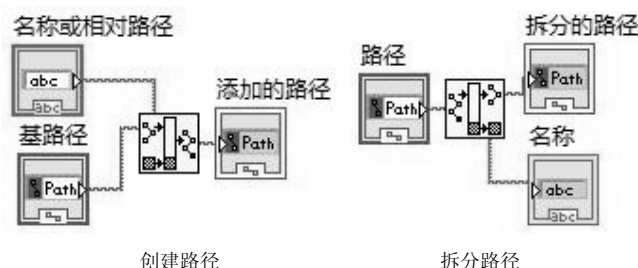


图 9.14 创建路径和拆分路径的连线板

这是一对功能相反的函数。创建路径函数用于从“基路径”和“名称或相对路径”合成“添加的路径”，例如，基路径为“C:\WINDOWS”，文件名为“abc.txt”时合成的新路径为：“C:\WINDOWS\abc.txt”。

拆分路径函数则用于从原路径中提取上一级目录“拆分的路径”和文件“名称”。

## 10. 文件常量

使用文件常量选板中的节点与文件 I/O 函数及 VI 配合使用，文件常量选板如图 9.15 所示。



图 9.15 文件常量选板

- 路径常量：使用路径常量在程序框图中提供一个常量路径。

- 空路径常量：该节点返回一个空路径。
- 非法路径常量：返回一个值为非法路径的路径。当发生错误而不想返回一个路径时，可以使用该节点。
- 非法引用句柄常量：该节点返回一个值为非法引用句柄的引用句柄。当发生错误时，可使用该节点。
- 当前 VI 路径：返回当前 VI 所在文件的路径。如果当前 VI 没有保存过，将返回一个非法路径。
- VI 库：返回当前所使用 VI 库的路径。
- 默认目录：返回 more 内目录的路径。
- 临时目录：返回临时目录路径。
- 默认数据目录：所配置的 VI 或函数所产生数据存储的位置。

## 9.2 文件 I/O 实例

下面举一例综合说明文件 I/O 相关函数的用法。

### 例 9-2 对电子表格文件的读写实例。

新建名为“对电子表格文件的读写.vi”的 VI，并在前面板上添加三个按钮，标签分别为“退出”、“写入”、“读取”。在框图中建立基于事件结构的框架，并添加对上述三个按钮的“值改变”事件响应代码，如图 9.16 至图 9.18 所示。

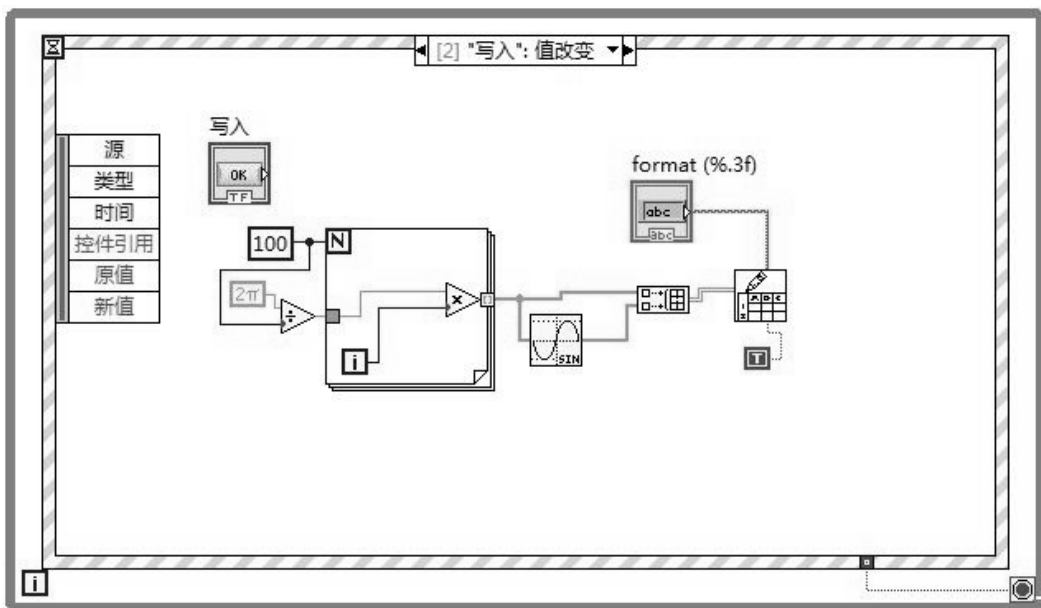


图 9.16 “写入”按钮的值改变事件响应代码

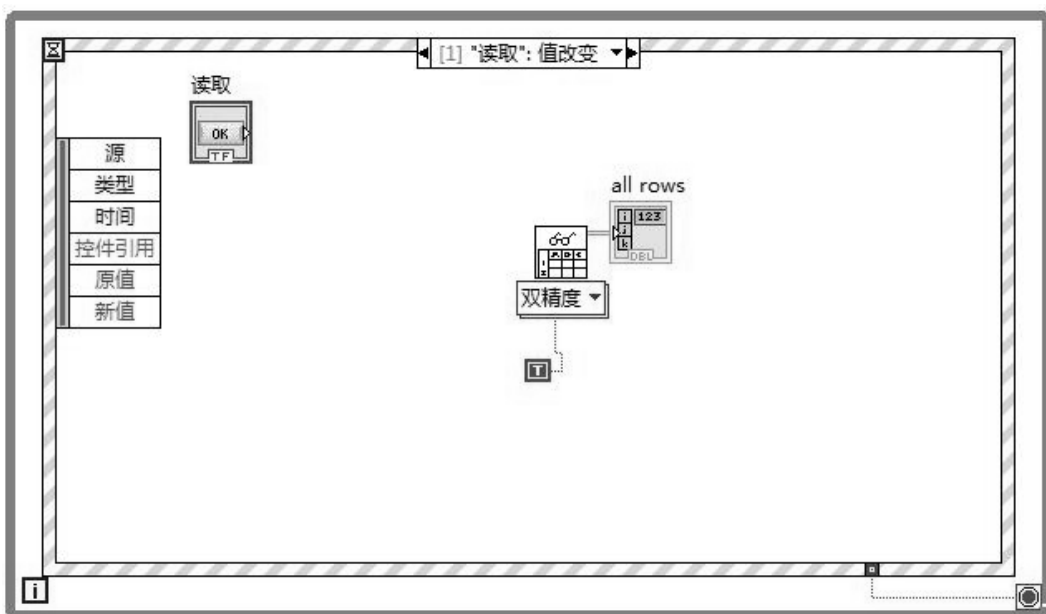


图 9.17 “读取”按钮的值改变事件响应代码

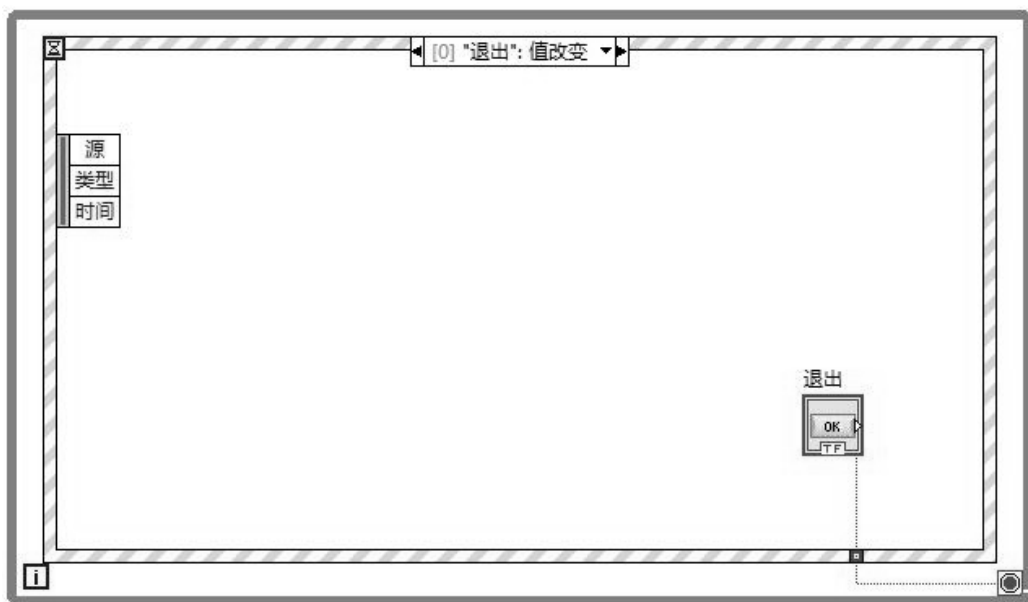


图 9.18 “退出”按钮的值改变事件响应代码

其中“写入”部分的代码利用 Write To Spreadsheet.vi 将一个正弦波信号的横、纵轴数据按照转置形式写入电子表格文件，“读取”部分的代码则可从电子表格文件中读出以前存入的数据。同样，也需指定按照转置形式读出才能得到正确形式的数据数组。

运行程序，单击“写入”按钮写入数据到某个文件，然后再单击“读取”按钮从该文件中读出数据，运行效果应如图 9.19 所示。



图 9.19 对电子表格的读写的效果

# 第 10 章 属性节点和方法节点

## 10.1 属 性 节 点

LabVIEW 中的一切对象（不仅包括输入控件和显示控件，也包括变量、前面板、VI、项目甚至应用程序本身）皆有属性，所谓属性是指 LabVIEW 预定义的用来描述对象状态的数据。改变对象的属性将改变对象相应的状态。

例如，大多数控件所具有的“可见（Visible）”属性，是一个布尔型属性，可以用来控制该控件在前面板上是否可见。

- 通过设置波形图表或波形图的相关属性，可以设置几乎一切绘图效果，如曲线颜色、线型、横纵轴坐标范围、光标位置等。
- 设置前面板的位置和大小相关属性，可以在程序运行时动态地改变程序窗口的位置和大小，创造出动态窗口的效果。

类似的例子举不胜举，正是因为 LabVIEW 为所有用户可能接触到的对象预定义了非常丰富的属性，才使得开发者在编写 LabVIEW 程序时具有很强的灵活性，几乎能够实现一切可以想象到的丰富效果，从而创建出高水平的程序和专业化的界面。

在框图中创建一个属性节点，一般可以使用两种方法。

- 通用方法：从函数选板中拖放属性节点到框图中，然后在其右键菜单中选择“选择类”下的子菜单项，选择关联类别，如控件、应用程序或 VI 等，然后单击属性节点中央，从弹出的菜单中选择具体属性。属性节点在函数选板中的位置如图 10.1 所示。
- 快捷方法：对于输入控件或显示控件，属性节点最快捷的创建方法是在其右键菜单中选择“创建-属性节点”，然后在列出的菜单中选择所需要的属性。

生成的属性节点以与原控件相同的标签名命名，表示属于原控件，属性节点中央为属性名。如果需要关联其他控件，可以在属性节点的右键菜单中选择“链接至”子菜单，其中列出了目前 VI 中所有的控件名，从中选择控件名就可关联对应的控件。

属性节点的两侧有错误信息簇端子，可以与其他属性节点或者其他具有错误信息簇端子的 VI 相连接，利用这一特性可以跟踪异常情况和引导数据流，允许的情况下应尽量使用。

例如，对一个按钮分别创建读和写两种状态的可见属性节点，并连接恰当的变量或常量，如图 10.2 所示。

若要对已有的属性节点创建一个副本，必须使用克隆操作，而不是简单的 Ctrl+C、Ctrl+V 复制操作。

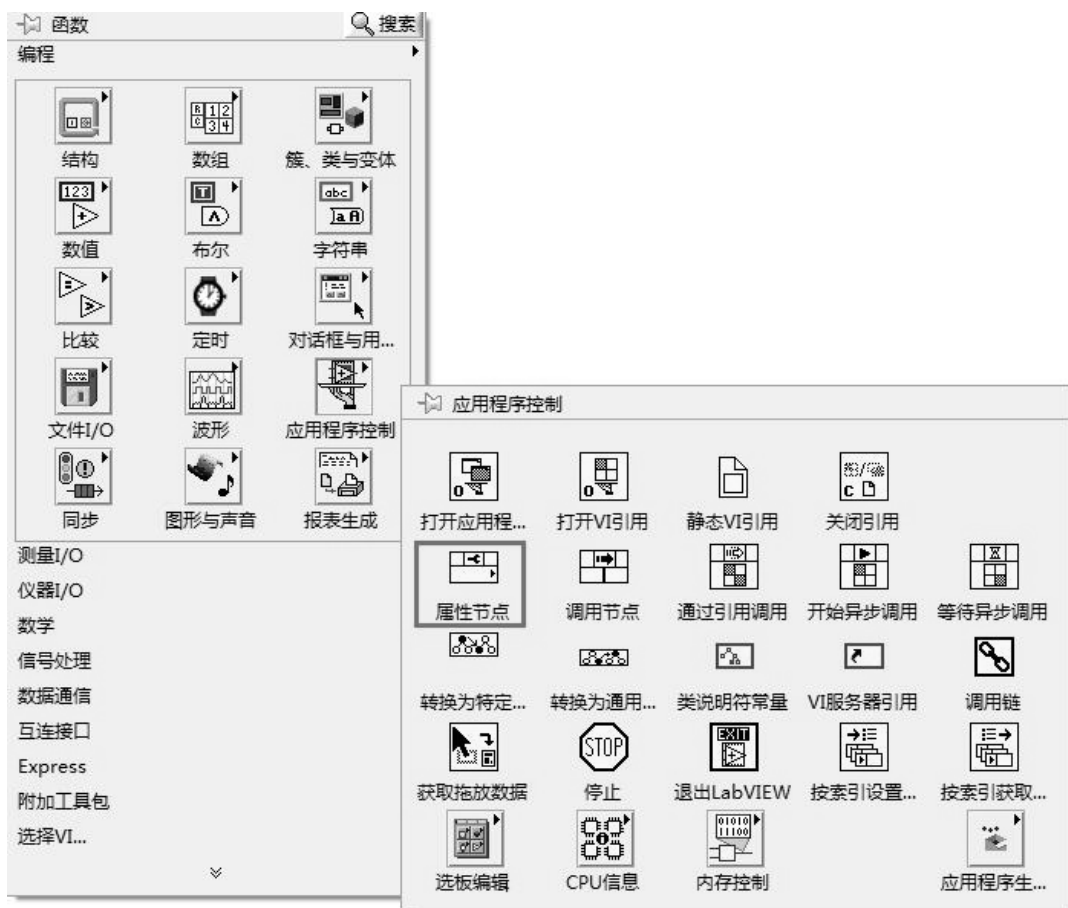


图 10.1 属性节点在函数选板中的位置

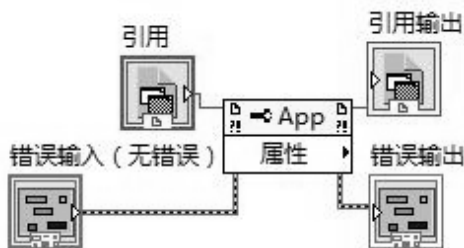
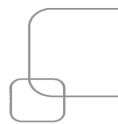


图 10.2 属性节点的连线板

属性节点的中央为属性名，如果需要修改为其他属性，用鼠标单击属性名并从弹出的列表中重新选择其他属性即可。由于大多数属性具有既可读取也可写入的特点，所以对应的属性节点有读和写两种状态，读状态属性节点中的小三角端子在右侧，而写状态属性节点中的小三角端子在左侧。

通过鼠标右键菜单中的“转换为读取”或“转换为写入”可以进行切换。但是对于某些只允许读取的属性，其对应的属性节点为只读型，“转换为写入”项就不可使用了，同样，对于只写型的属性，“转换为读取”项不可用。





默认一个属性节点只对一个属性进行读或写操作，实际上一个属性节点还可以支持对多个属性进行操作，添加新属性的方法是用鼠标拖住属性节点的下边沿，向下拖动以露出新属性。也可以通过右键菜单中的“添加元素”命令添加新属性，或者“删除元素”命令删除某个属性。

多个属性共用一个属性节点可以节省占用框图的空间，也便于阅读，但各个属性之间是相互独立的，实际执行顺序为按照在节点中的排列顺序从上到下。

使用右键菜单中的“全部转换为读取”或“全部转换为写入”命令可以将该节点中的所有属性都设置为读或写状态，也可以通过每个属性自己右键菜单中的“转换为读取”或“转换为写入”对每个属性分别设置不同的读或写状态。

下面对输入控件和显示控件最常用的 6 个属性及使用时的注意事项进行介绍。

### 1) 可见 (Visible)

LabVIEW 中的控件虽然不能动态地创建或删除，但却可以动态地显示或隐藏，可见属性专门用于在前面板上显示或隐藏控件，它是一个既可读也可写的布尔型数据，其值的真或假代表控件显示或隐藏。

### 2) 值 (Value)，值 (信号) [Value (Signaling)]

值属性是与控件所代表变量同一类型的数据，既可读也可写，与控件当前值相等，用户可以使用这一属性动态地对变量值进行读或写，这与局部变量的用途相仿。

值 (信号) 属性的作用也类似，但只用于写入变量值，不可读取，在写入变量的同时会产生一个“值改变”事件，用户在编程时可利用此事件编写响应方法，其具体使用方法在后面章节中再介绍。

因为与局部变量的用途相仿，所以值和值 (信号) 属性也不可用于机械动作设置为 Latch 的按钮，否则它们的使用会影响变量值，LabVIEW 对此类错误会自动检出并报错。

### 3) 禁用 (Disabled)

禁用属性用于表示控件的用户使能/禁用状态，这是一个取值范围为 0、1 或 2 的整数型数据，既可读也可写，默认值为 0，这些取值的具体含义如下：

- 0——控件可以正常使用。
- 1——控件被禁用，用户无法使用该控件进行操作，但前面板上的外观效果仍然正常。
- 2——控件被禁用，且外观变为灰色。

在某些情况下，可能希望某些控件对应用程序的使用者暂时失效，以避免误操作，这时将禁用属性设置为 1 或 2 就可以禁用该控件，直至再次将该属性设置为 0 为止。

### 4) 闪烁 (Blinking)

闪烁属性为既可读也可写的布尔型数据，其值的真或假用于表示控件是否在前面板上闪烁。对于希望用户注意到的提示信息，如出错提示、报警提示等信息，可以将相应控件设置为闪烁，引起用户注意。



### 5) 位置 (Position)

位置属性用于表示控件在前面板上相对于坐标原点的位置。位置属性是一个簇类型的数据，既可读也可写。簇元素由两个整型变量左 (Left) 和上 (Top) 组成，分别表示控件最左上角点与坐标原点横、纵轴的相对坐标偏移量，分别以向右和向下为正方向。这两个元素也可以通过子属性位置：居左 (Position: Left) 和置顶 (Position: Top) 独立进行设置或读取。

所谓前面板的坐标原点是指新建 VI 之后尚未用滚动条滑动前面板窗口时，前面板的最左上角点以一个灰色圆点表示，稍稍滑动窗口滚动条就可以露出该点。

位置属性用于动态移动控件在前面板上的位置，使用时需注意，所谓控件最左上角点是指将控件主体及所有显示的附件（如标签、标题等）都包含在内计算得到的。

### 6) 边界 (Bounds)

边界属性用于表示控件的边界范围，它是一个簇类型数据，但只可读不可写。簇元素由高度 (Height) 和宽度 (Width) 两个整型变量组成，分别用来表示控件边界的高度和宽度，并可以分别独立进行设置。

这里所指的边界范围也是指将控件主体及所有显示的附件都包含在内的总体范围，与调整对象大小窗口中的尺寸是不一样的，后者仅是指控件主体的尺寸。

**例 10-1** 利用按钮的“可见”属性来控制按钮是否可见。

(1) 启动 LabVIEW 2013，打开前面板，选择控件选板中的“新式→布尔→开关按钮”，在前面板中放置两个“开关按钮”。

(2) 切换到程序框图窗口，右击第二个“开关按钮”，选择“创建→属性节点→可见”，创建第二个“开关按钮”的“可见”属性，并用连线工具连线，如图 10.3 所示。未运行程序前的前面板如图 10.4 所示。

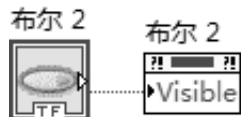


图 10.3 创建“可见”属性



图 10.4 程序运行前的前面板

(3) 切换到前面板，运行程序，结果如图 10.5 所示。

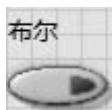


图 10.5 运行结果

**例 10-2** 编程实现列表框的编辑，并获取列表框中选中的输出选项的名称。

(1) 启动 LabVIEW 2013，打开前面板窗口，选择控件选板中的“编程→列表、表格和树→列表框”，将其置于前面板，右击“列表框”，选择“选择模式→0 或多项”来设置列表框模式。

(2) 切换到程序框图窗口，右击“列表框”，选择“创建→属性节点→显示项→显示符

合”，创建“列表框”属性，将各属性接口全部转换为读取，并创建常量，在“项名”中输入项目名；在“项符号”中通过“编程→对话框与用户界面→列表框符号下拉列表控件常量”选择“列表框符号下拉列表控件常量”，将其置于“项符号”常量中。此时，运行程序，在前面板列表框中就会显示所有的项目名，如图 10.6 所示。

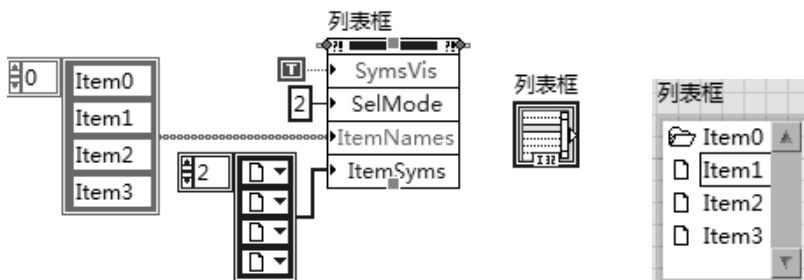


图 10.6 列表框的编辑

(3) 在函数选板中选择“编程→结构→For 循环”，在程序框图中创建 For 循环结构体，选择“编程→数组→索引数组”，将其置于 For 循环体中，并在循环体中创建列表框“项符号”属性。

(4) 切换到前面板，选择“新式→数组、矩阵与簇→数组”控件，在前面板中创建两个空数组，选择“新式→数值→数值显示控件”和“新式→字符串与路径→字符串显示”控件，置于两个空数组中。切换到程序框图中连线，如图 10.7 所示。

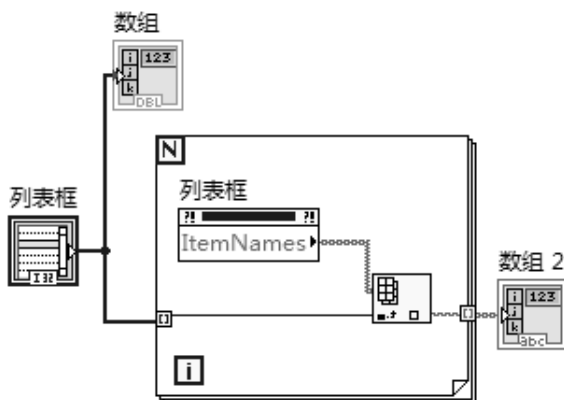


图 10.7 在程序框图中连线

(5) 再次切换到前面板，左键选中项目 2 和项目 3，运行程序，在第一个数组中显示项目序号，在第二个数组中显示项目名，运行结果如图 10.8 所示。



图 10.8 程序运行结果

## 10.2 调用节点

调用节点 (Invoke Node) 和属性节点非常类似, 调用节点就好比控件的一个函数, 它会执行一定的动作, 有些时候还需要输入参数或返回数据。

调用节点和属性节点的调用方法完全一样, 也有两种办法。一种是右击控件并选择“创建→调用节点 (Create|Invoke Node)”选项, 另一种办法是在“编程→应用程序控制 (Programming|Application Control)”面板上选择“调用节点”。如果需要关联到其他类别, 可以从调用节点的右键菜单中“选择类”子菜单下选择。若需关联到其他控件, 可以在右键菜单“链接至”子菜单下, 选择其他控件名就可以关联到对应的控件。

例如, 对一个波形图表控件创建的“获取图像 (Get Image)”调用节点, 如图 10.9 所示。



图 10.9 调用节点

生成的调用节点类似于一个函数节点, 同样具有输入/输出参数, 用户按照帮助文档中对该方法的说明指示, 将参数连接到合适的变量或常量, 就可以正确调用该节点对应的方法。调用节点所具有的错误信息簇端子也可以用来引导数据流和处理异常。

## 10.3 实 例

### 例 10-3 获取控件前面板图像实例。

获取图像 (Get Image) 是多数控件都具有的调用节点方法, 也是经常使用的方法之一, 用来获取控件在前面板上的图像, 它具有以下输入/输出参数:

- Image Depth——输入参数, 图像颜色位数, 位数越多可表现的颜色越多, 1 位只能表示黑白两色, 4 位可表示 16 色, 8 位可表示 256 色, 24 位可表示 16 777 216 色, 又称真色彩。
- BG Color——输入参数, 图像背景色。
- Image Data——输出参数, 获得的图像数据, 可以连接至相关 VI 保存为图像文件。

利用“获取图像”调用节点获取一个波形图的前面板图像, 步骤如下:

(1) 新建一个 VI, 添加 While 循环结构、波形图显示控件, 以及信号生成 VI “混合单频与噪声 (Tones and Noise.vi)”, 用来在波形图中显示带噪声的正弦波形。波形图的标签和

曲线图例设为隐藏。

(2) 对波形图显示控件创建调用节点，选择为“获取图像”，输入参数分别连接为 24 位透明色，输出参数连接至 VI “写入 PNG 文件 (Write PNG File.vi)”，用来保存图像为 PNG 格式文件。

这部分保存代码添加在一个条件结构中，选择条件用按钮进行控制，当按钮按下时才进行保存。

(3) 得到的控件前面板图像如图 10.10 所示，运行该程序，并单击“保存”按钮，在对话框中选择路径和写入文件名，即可保存波形图的当前前面板图像。

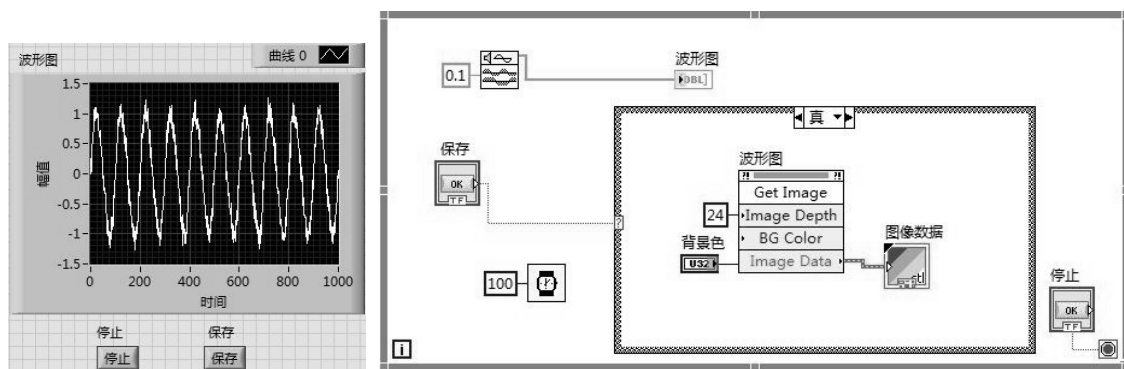


图 10.10 获取控件前面板图像

属性节点与调用节点的使用方法是类似的，关键还是需要理解清楚具体属性或方法的含义，结合实际需要灵活恰当地运用。

LabVIEW 中除了为对象预定义属性和方法外，从 8.0 版本开始允许用户对自定义控件自行定义属性和方法，是使用 XControl 实现的，可以看作对面向对象编程方法的初步支持，感兴趣的读者可自己查阅相关资料进行学习。

# 第11章 数学计算

数字信号处理、信号与系统是电子信息工程和通信类专业的重要专业基础，而且随着电子信息技术的飞速发展，它们在信息技术中的地位越来越重要。但是由于该技术的学习内容较复杂，公式、概念繁多，理论性与实践性要求都较高，学习过程中往往缺乏有效的计算机辅助平台，学生不易掌握。

学习好数字信号分析与处理，行之有效的办法就是在理解了基本理论的基础上多动手做相关的实验，主动验证相关结论或设计新的算法，自主分析实验结果，从而达到全面理解数字信号处理理论的目的。

LabVIEW 作为自动化测试、测量领域的专业软件，其内部集成了 600 多个分析函数，用于信号生成、频率分析、概率、统计、数学运算、曲线拟合、插值、数字信号处理等各种数据分析应用。另外，LabVIEW 还提供了附加工具软件专用于某些信号处理，如声音与振动、机器视觉、RF/通信测量、瞬态/短时持续信号分析等。此外，LabVIEW 还能与 MATLAB 联合编程，从而实现更为强大的编程功能。

与其他编辑语言或平台相比，LabVIEW 尤其适合数字信号分析与处理的学习和相关实验设计，这是因为它具有以下其他平台无可比拟的优势。

- 具有良好的图形显示功能，提供种类齐全的各种波形图和波形图表控件，能够以多样化的方式直观显示各种信号波形。
- 图形化的编辑方式，学习门槛较低，易于掌握，省去了许多烦琐的编程细节，能够使用户专注于数字信号处理算法的研究和设计，而这些正是相关课程的精髓。
- 拥有数量众多、功能齐全的各种信号分析与处理 VI，供用户随意调用，从而组合实现所需要的信号处理功能，对于各种应用需求，无论是课程学习还是工程设计都能够满足；
- 具有良好的扩展性，无论是通过附加工具包扩展，还是通过与其他平台（如 MATLAB/Simulink）的接口扩展，都可很方便地进一步扩展其信号处理功能。

使用 LabVIEW 编写和设计数字信号处理程序，主要靠其提供的种类丰富的各种信号处理 VI 实现，包括信号生成、调制、滤波、频谱分析、变换等各种算法都有相关可调用的 VI 库，这些 VI 库的位置位于函数选板“信号处理”下。

LabVIEW 作为图形化的开发语言，与传统的文本编程语言有很大区别。如果读者已经能够熟练通过传统的文本语言实现数学分析程序，可能会质疑 LabVIEW 写数学算法的能力。因为 LabVIEW 是通过连线和框图的方式编程，复杂的算法会不会导致繁杂的连线呢？的确，如果仅以 LabVIEW 的基本运算符号和程序结构来实现复杂的算法，很可能导致繁杂的连线。针对这一点，LabVIEW 封装了大量的数学函数以实现数学分析，并提供了基于文本编程语言的公式节点和 MathScript。通过这些封装好的 VI 函数并结合公式节点或

MathScript，程序框图可以变得非常简洁，用户可以把精力集中在需要解决的问题上，而不必再为数学算法费心。数学分析 VI 函数最底层是通过 C 语言实现的，因此具有很高的运行效率。此外，由于图形化编程时独立于平台，不会因为不同平台由于数据类型不同而导致重新编辑或编译程序。

因此通过 LabVIEW 实现数学分析不仅不会导致繁杂的连线，反而由于采取了图形化编程和文本编程相结合的方式，它比单纯的文本编程语言具有更大的优势。此外，由于 LabVIEW 能方便地与各种数据采集设备直接连接，用户可以直接将采集到的数据进行数学分析，这也是 LabVIEW 的巨大优势之一。

LabVIEW 所提供的数学分析函数位于函数选板的“数学”子选板下，如图 11.1 所示。

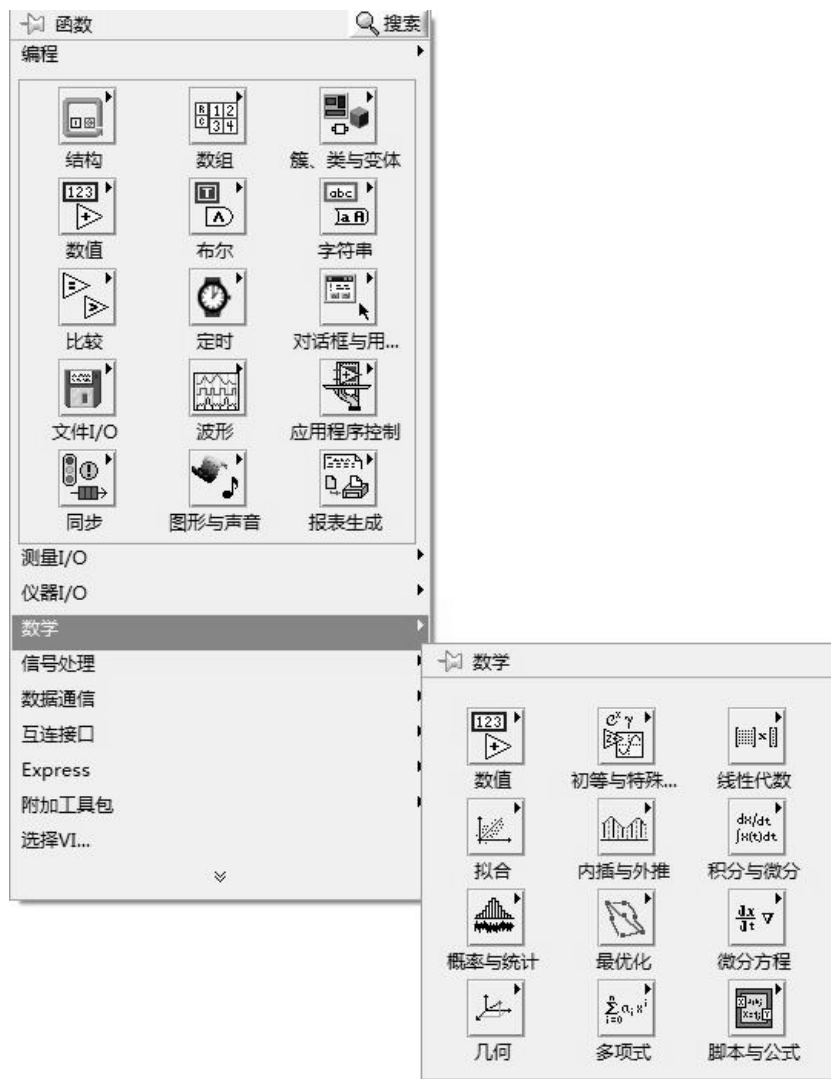


图 11.1 数学分析 VI 函数选板

数学分析 VI 函数库分为 12 个部分，其具体功能如表 11.1 所示。

表 11.1 数学分析 VI 函数各子面板功能列表

名 称	功 能
数值/Numeric	最基本的数学操作，例如，加减乘除、类型转换、数据操作等
初等与特殊函数/Elementary&Special Functions	一些常用的数学函数，例如，正余弦函数、指数函数双曲线函数、离散函数和贝塞尔函数等
线性代数/Linear Algebra	线性代数，主要是与矩阵操作相关的函数
拟合/Fitting	曲线拟合和回归分析
内插与外推/Interpolation&Extrapolation	一维和二维的插值函数，包括分段插值、多项式插值和傅里叶插值
积分与微分/Intergration&Differentiation	积分与微分函数
概率与统计/Probability&Statistics	概率与统计
最优化/Optimization	最优化
微分方程/Differential Equations	解常微分方程
几何/Geometry	几何
多项式/Polynomial	多项式计算和分析
脚本与公式/Scripts&Formulas	脚本节点、公式节点和公式解析的相关函数

由于数学分析共有数百个函数，本章将主要介绍每一类函数的应用方程，并挑选其中具有代表性的函数进行举例说明。当读者在实际应用中需要使用某一函数时，可以查询 LabVIEW 的联机帮助。

## 11.1 基本数学函数

函数选板的“数学（Mathematics）→初等与特殊函数（Elementary&Special Functions）”面板下包含了常用的基本数学函数，如图 8.2 所示。该面板将常用的数学函数分为 12 类：三角函数、指数函数、双曲线函数、门函数、离散数学、贝塞尔函数、Y 函数、超几何分布函数、椭圆积分函数、指数函数、误差函数和椭圆抛物线函数。

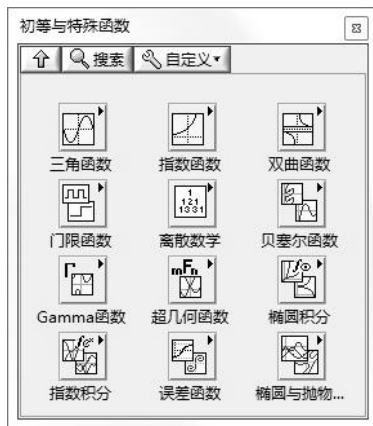


图 11.2 基本数学函数面板



## 11.2 线性代数

线性代数在现代工程和科学领域中有着广泛的应用，因此 LabVIEW 也提供了强大的线性代数运算功能。线性代数函数面板位于函数选板的“数学（Mathematics）→线性代数（Linear Algebra）”面板下，如图 11.3 所示。

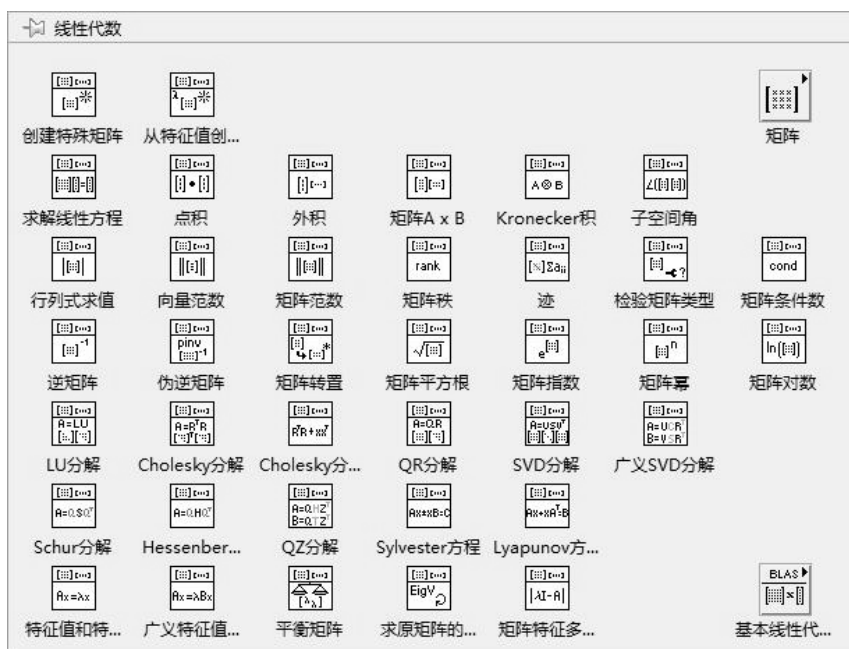


图 11.3 线性代数函数面板

下面通过解线性方程组来举例说明线性代数运算函数的用法。线性方程组的 VI 函数为“求解线性方程（Solve Linear Equations.vi）”。

例 11-1 解如下线性方程组  $Ax=b$ ，其中：

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

(1) 新建一个 VI，添加“求解线性方程”函数，矩阵输入控件  $A$ 、 $b$ ，矩阵输出控件  $x$ ，然后连线。

(2) 将数据输入数组，运行得到的程序前面板和框图如图 11.4 所示：

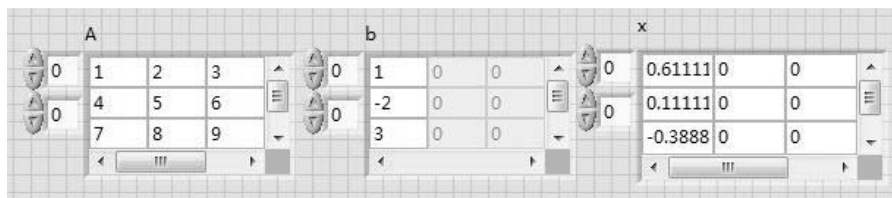


图 11.4 解线性方程组

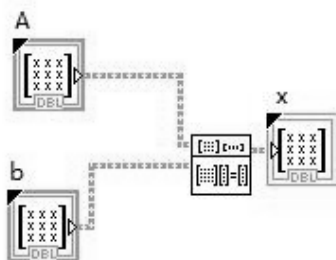


图 11.4 解线性方程组 (续)

即最后得出  $x$  的值为:

$$x = \begin{bmatrix} 0.61 \\ 0.11 \\ -0.39 \end{bmatrix}$$

LabVIEW 也支持复数矩阵, 与实数矩阵不同的是, 这里必须使用复数矩阵控件, 还需要将输出的数组转换为复数矩阵, 如图 11.5 所示是解线性方程组。

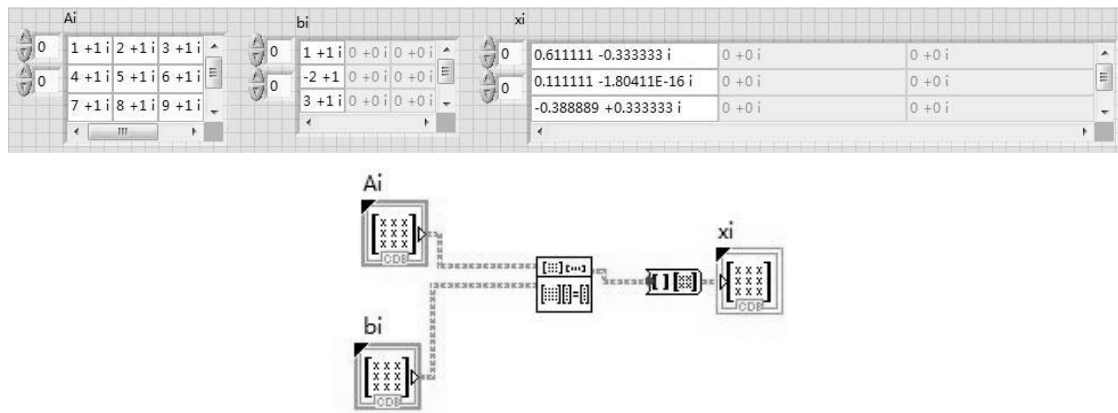


图 11.5 解复线性方程组

从以上例子可以看出, 通过 LabVIEW 实现线性代数运算的代码非常简单。用户可以把精力集中在所做的数学运算上, 而不需要去考虑数据类型的定义等问题。

## 11.3 曲线拟合

曲线拟合在分析实验数据时非常有用, 它可以从大量的离散数据中抽象出内部规律。LabVIEW 包含了大量的曲线拟合函数以满足不同的拟合需要, 其中不仅包括二维曲线拟合, 还包括三维曲线拟合, 曲线拟合函数面板位于函数面板的“数学(Mathematics)→拟合(Fitting)”面板中, 如图 11.6 所示。

对数据进行二维曲线拟合时, 需要输入数据的坐标  $(x_i, y_i)$ , 即  $X$  数组和  $Y$  数组。曲线拟合的目的就是找出  $x_i$  和  $y_i$  的函数关系  $y=f(x)$ 。对于不同的对象有不同的拟合方法。如曲线拟合 (Linear Fit)、指数拟合 (Exp Fit)、幂函数曲线拟合 (Power Fit)、高斯曲线拟合

(Gauss Peak Fit)、对数拟合 (Logarithm Fit)、非线性曲线拟合 (Nonlinear Curve Fit) 等。



图 11.6 曲线拟合函数面板

例 11-2 最小二乘法曲线拟合举例。

利用最小二乘法拟合曲线，因变量  $y$  与自变量  $x$  的关系表达式为：

$$y = f(a, x) = \sum_{i=0}^n a_i f_i(x) = a_0 f_0(x) + a_1 f_1(x) + \dots + a_n f_n(x)$$

本例中，原始数据为原始函数叠加一定的噪声产生，即

$$y = \sin(x^2) + 3\cos(x) + \frac{4x}{x+1} + \text{Noise}$$

其中：

$$f_0(x) = 1;$$

$$f_1(x) = \sin(x^2);$$

$$f_2(x) = 3\cos(x);$$

$$f_3(x) = \frac{x}{x+1};$$

$$f_4(x) = x^4。$$

下面通过广义线性拟合函数 General LS Linear Fit.vi 来求解回归函数，其图标和端口如图 11.7 所示。

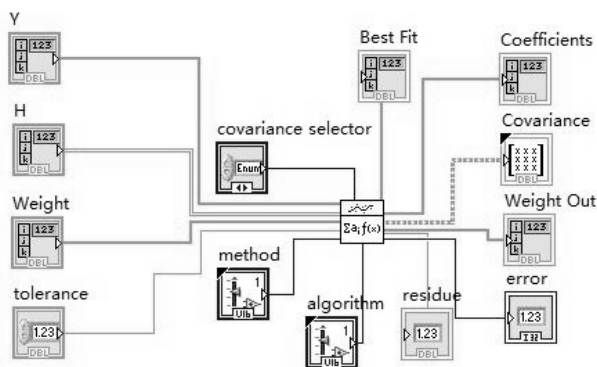


图 11.7 广义线性拟合函数的图标和端口

其中,  $Y$  为原始数据,  $H$  矩阵是根据猜测函数产生的, 它是猜测函数在自变量各点的函数值, Best Fit 为拟合曲线的  $Y$  值, Coefficient 为回归系数。通过该函数实现的曲线拟合程序如图 11.8 所示。

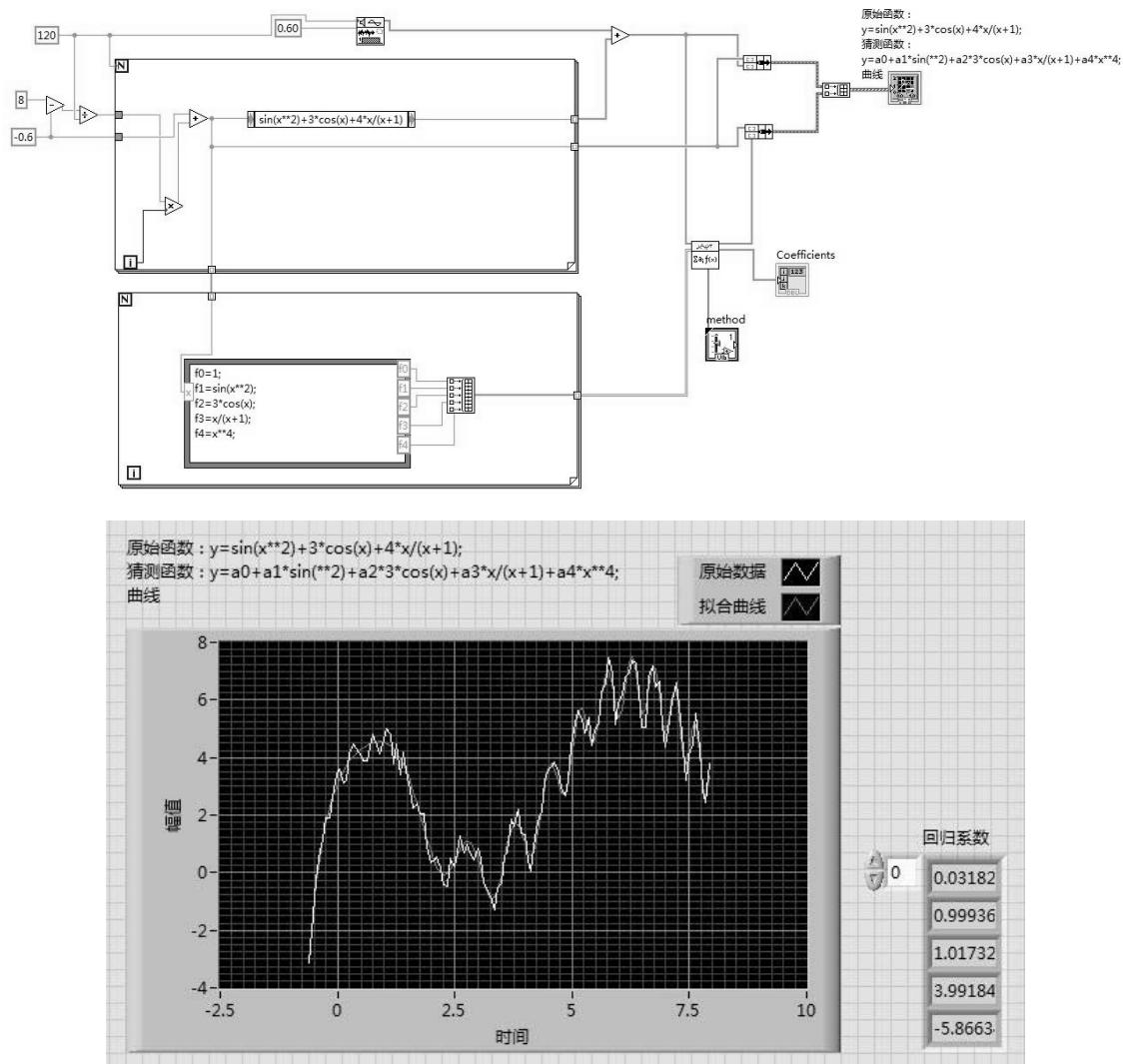


图 11.8 最小二乘法曲线拟合

## 11.4 插 值

插值是在离散数据之间补充一些数据, 使这组离散数据能够符合某个连续函数。插值是计算数学中最基本和最常用的手段, 是函数逼近理论中的重要方法。利用它可以通过函数在有限点处的取值情况估算该函数在其他点的值, 即通过有限的的数据得出完整的数学描述。

插值函数面板位于函数选板的“数学（Mathematics）→内插与外推（Interpolation & Extrapolation）”面板中，如图 11.9 所示。

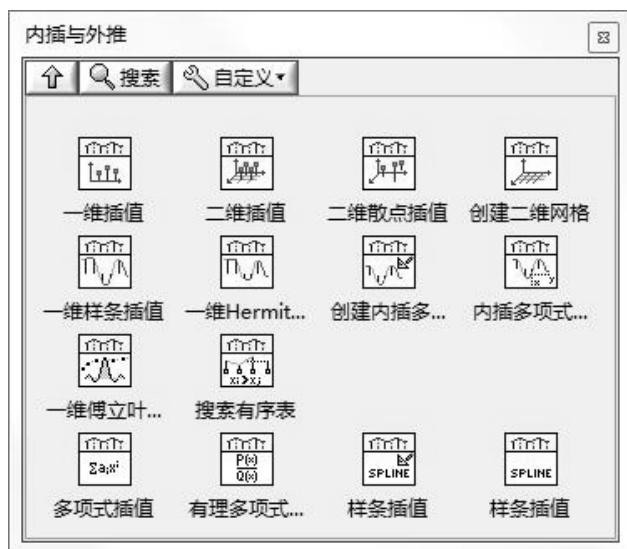


图 11.9 插值函数选板

内插与外推函数面板包含了多个插值函数：一维插值、二维插值、一维样条插值、一维 Hermite 插值、一维傅里叶插值等，所有函数均可以进行内插与外插。

LabVIEW 将各种插值算法分装为一个简单的 VI 函数“一维插值（Interpolate 1D.vi）”，如图 11.10 所示。

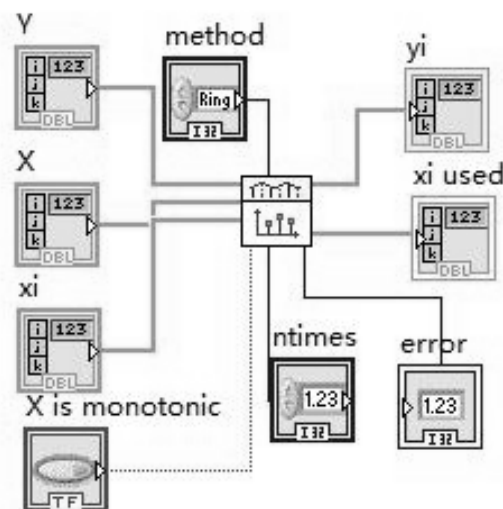


图 11.10 一维插值函数的图标和端口

其中，method 用于选择插值方法；Y 为原始函数值；X 为横坐标；xi 为插值点的横坐标；yi 为插值点的函数值；xi used 为使用到的插值点横坐标值。

## 11.5 数值积分与数值微分

数值积分与数值微分相对简单，这些函数位于函数选板的“数学（Mathematics）→积分与微分（Intergration&Differentiation）”面板下，如图 11.11 所示。



图 11.11 积分与微分函数选板

其中，数值积分用于计算一维、二维或三维数组的数值积分，有四种积分方法可以选择。

$$\text{设 } I_f = \int_0^1 f(x) dx = \sum_j \text{partialSums}$$

这四种方法对应的 PartialSums 的算法为：

$$\text{梯形公式 (Trapezoidal)} \quad \frac{1}{2}(x_i + x_{i+1}) dt$$

$$\text{Simpson 公式积分} \quad \frac{1}{3}(x_{2i} + 4x_{2i+1} + x_{2i+2}) dt, k=2$$

$$\text{Simpsons 3/8} \quad \frac{1}{8}(3x_{3i} + 9x_{3i+1} + 9x_{3i+2} + 3x_{3i+3}) dt, k=3$$

$$\text{Bode} \quad \frac{1}{45}(14x_{4i} + 64x_{4i+1} + 24x_{4i+2} + 64x_{4i+3} + 14x_{4i+4}) dt, k=4$$

积分  $x(t).vi$  用于计算函数的不定积分。计算公式为  $Y = \int(t) dt$ ， $Y$  对应于一维数组，

$y_i = \frac{1}{6} \sum_{j=0}^i (x_{j-1} + 4x_j + x_{j+1}) dt$ ，其中  $i=0,1,2,\dots,n-1$ 。 $n$  为输入数组的大小， $i=0$  时的  $x_{-1}$  值由初始条件决定， $i=n-1$  时的  $x_n$  值由终止条件决定。

求导  $x(t).vi$  用于计算函数的导数。假设  $F(t)$  的导数定义为  $f(t) = \frac{d}{dt} F(t)$ ，数组  $Y$  代表

$dx/dt$  的输出序列，则  $y_i = \frac{1}{2dt}(x_{i+1} - x_{i-1})$ ，其中  $i=0,1,2,\dots,n-1$ 。 $n$  为输入数组的大小， $i=0$  时的  $x_{-1}$  值由初始条件决定， $i=n-1$  时的  $x_n$  值由终止条件决定。

**例 11-3** 数值积分与数值微分举例。

设  $f(x) = e^{\sin x}$ ，求该函数在  $[0, \pi]$  上的定积分、不定积分和导数，如图 11.12 所示。

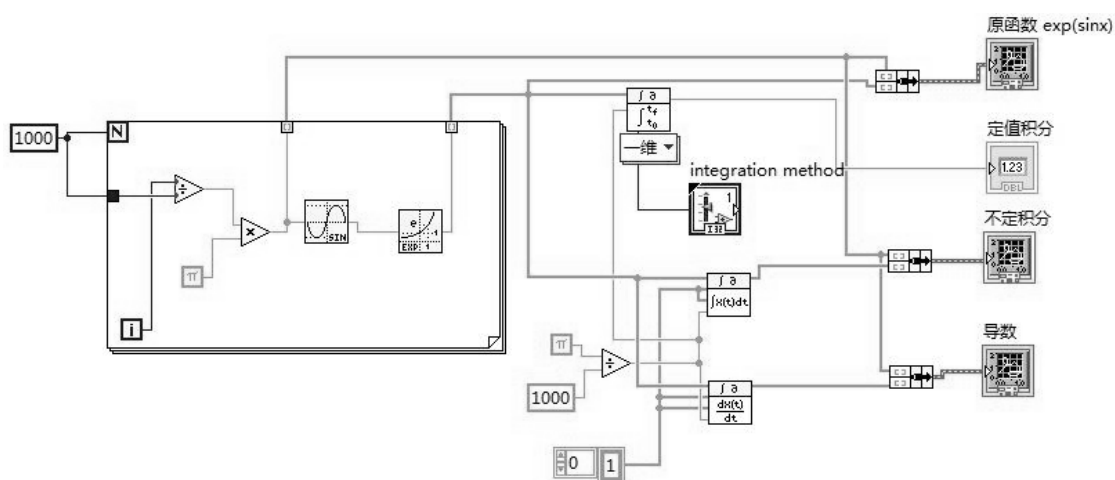
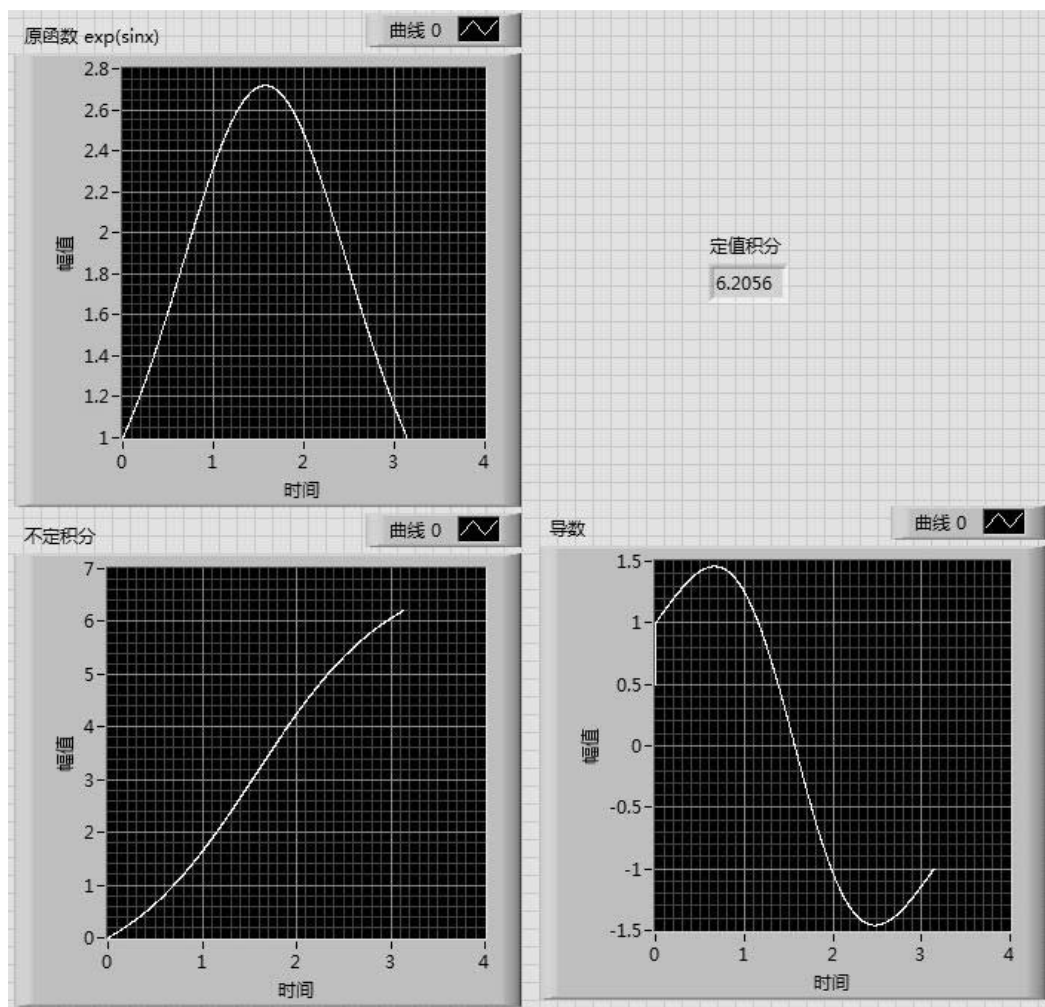


图 11.12 积分与微分函数实例

## 11.6 概率与统计

概率论和数理统计是研究和揭示随机现象统计规律的一门数学学科。随机性的普遍存在使人们发展出了多种数学方法用于揭示其内部规律。随着电子计算机的出现，计算机大批量、高速处理数据的能力使大量的数据分析成为可能。LabVIEW 也提供了大量的概率与统计函数，这些函数位于函数选板的“数学（Mathematics）→概率与统计（Probability & Statistics）”面板下，如图 11.13 所示。

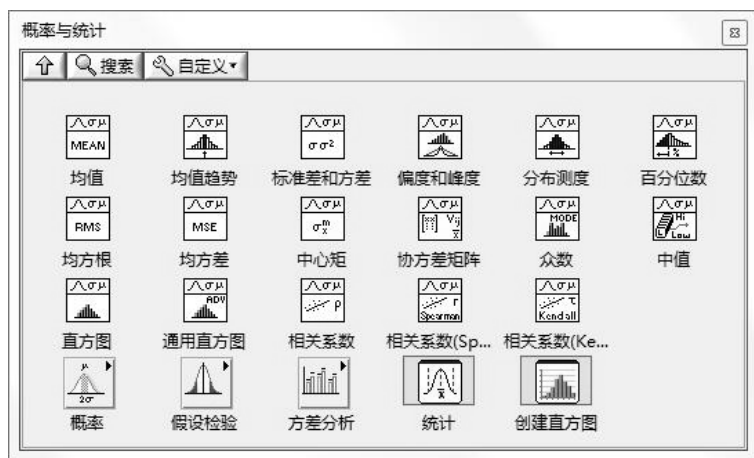


图 11.13 概率与统计函数面板

由于用于概率和统计的函数较多，这里只介绍其中两个 Express VI 函数：统计（Statistic）和创建直方图（Create Histogram）。这两个函数能满足大部分基本的概率统计需求。其中统计函数可以对大部分的基本统计参数进行计算，如算数平均值、中数、均方根和标准方差等。创建直方图函数可以对数据进行柱状图统计。

### 例 11-4 概率与统计函数举例。

在本例中，首先通过高斯白噪声（Gaussian White Noise.vi）函数产生一个满足高斯分布的随机数序列，然后通过创建直方图（Create Histogram）和统计（Statistic）两个 Express VI 对该随机序列进行分析，所得前面板和程序框图如图 11.14 所示。

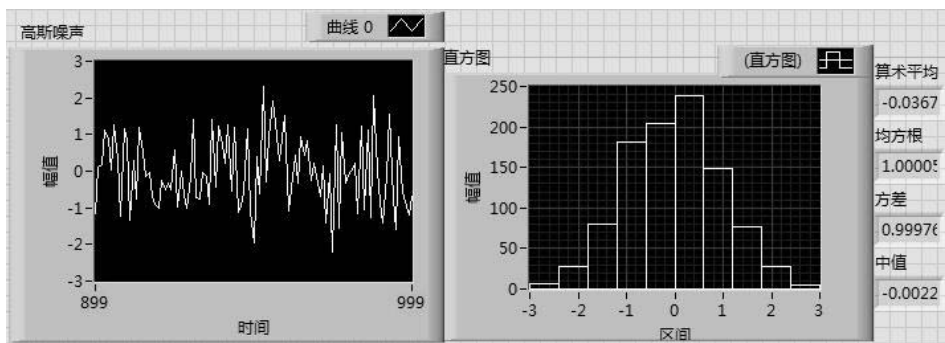


图 11.14 对高斯噪声进行统计分析实例



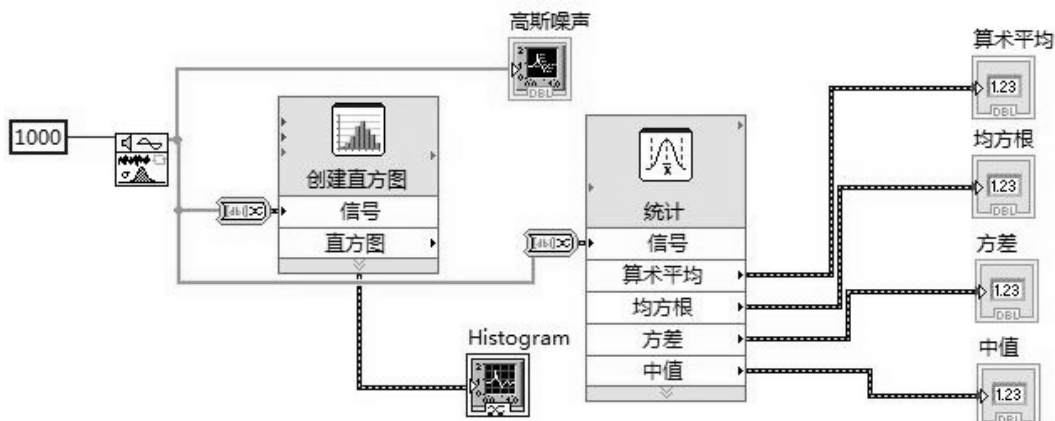


图 11.14 对高斯噪声进行统计分析实例（续）

## 11.7 最优化

最优化是一门古老而又年轻的学科，它的起源可以追溯到法国数学家拉格朗日关于一个函数在一组等式约束条件下的极值问题。如今这门学科在工业、军事技术和管理科学等各个领域均有着广泛的应用，并发展出组合优化、线性规划、非线性规划、动态控制和最优控制等多个分支。最优化问题的一般形式为：

$$\min f(x), \text{ s.t. } x \in \Omega.$$

即求目标函数  $f(x)$  的极值，约束条件为  $x \in \Omega$ ， $\Omega$  为可行解域。

LabVIEW 中最优化相关的 VI 函数位于函数选板的“数学（Mathematics）→最优化（Optimization）”函数面板下，如图 11.15 所示。



图 11.15 最优化函数面板

这些函数基本上涵盖了各种最优化问题，其中线性规划单纯形法（Linear Programming Simplex Method.vi）用于解线性规划方程，无约束最优化（Unconstrained Optimization.vi）用

于解无约束优化问题，一元函数局部最小值（Brent 法）（Brent with Derivatives 1D.vi）用于在给定区间计算一元函数的局部极小值点，一元函数局部最小值（黄金分割法）（Golden Selection 1D.vi）用黄金分割法计算一元函数的极小值点，多维共轭梯度法（Conjugate Gradient nd.vi）用共轭梯度法计算  $n$  元函数的局部极小值点，一元函数的所有最小值（Find all Minima 1D.vi）和多元函数的所有最小值（Find all Minima nD.vi）分别用于计算一元和  $n$  元函数在给定区间内的所有极值点，Chebyshev 逼近（Chebyshev Approximation.vi）用 Chebyshev 多项式逼近给定函数。

下面通过举例来说明如何实现现实中的最优化问题。

#### 例 11-5 最优化问题举例。

对边长为 4m 的正方形铁板，在四个角上剪去相等的正方形以制成方形无盖水槽，应该如何剪才能使所得的水槽容积最大？

解：设剪去的正方形的边长为  $x$ ，则水槽的容积为  $V = (4-2x)^2 x$ 。

建立无约束优化模型为  $\max y = (3-2x)^2 x$ ， $0 < x < 2$ 。

把求极大值的问题转化为求极小值的问题，即  $\min y = -(3-2x)^2 x$ ，因此利用“一元函数的所有最小值（Find all Minimal 1D.vi）”函数即可找到本问题中的一维函数在  $[0, 2]$  上的最小值，所得前面板和程序框图如图 11.16 所示。

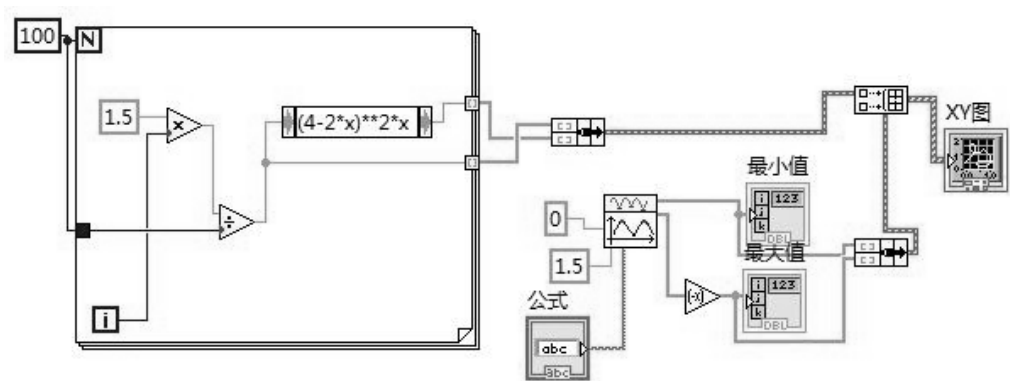
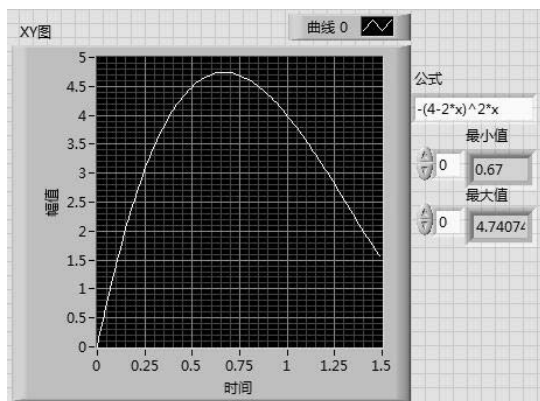


图 11.16 最优化问题举例

由程序结果可得，当  $x = 0.67$  时水槽的容积最大，此时水槽的容积约为  $4.74\text{m}^3$ 。

## 11.8 常微分方程

解常微分方程在工程计算中经常用到，通过解常微分方程可以解决很多几何、力学和物理学等领域的各种问题。LabVIEW 提供了多个 VI 函数用于解常微分方程，这些函数位于函数选板的“数学（Mathematics）→微分方程（Differential Equations）”面板下，如图 11.17 所示。

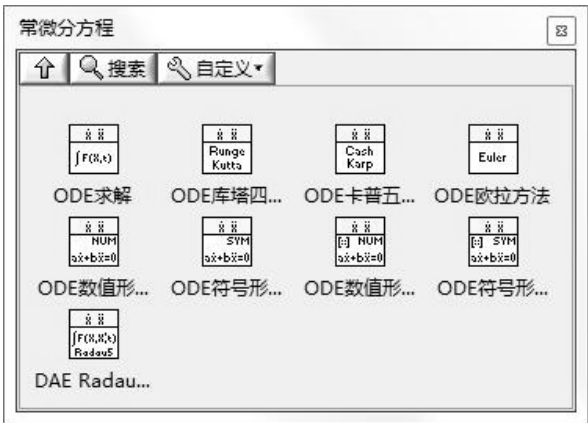


图 11.17 “常微分方程”面板

这些函数的功能如表 11.2 所示。

表 11.2 常微分方程函数功能列表

名 称	功 能
ODE 求解/ODE Solver.vi	解带初值的常微分方程 $X' = F(X, t)$
ODE 库塔四阶方法/ODE Runge Kutta 4th Order.vi	用龙格-库塔方法解带初值的常微分方程
ODE 卡普五阶方法/ODE Cash Karp 5th Order.vi	用 Cash Karp 方法解带初值的常微分方程
ODE 欧拉方法/ODE Euler Method.vi	用欧拉方法解带初值的常微分方程
ODE 数值形式线性 $n$ 阶微分方程 /ODE Linear nth Order Numeric.vi	用数值解法解 $n$ 阶线性齐次微分方程
ODE 符号形式线性 $n$ 阶微分方程 /ODE Linear nth Order Symbolic.vi	用符号解法解 $n$ 阶线性齐次微分方程
ODE 数值形式线性微分方程/ODE Linear System Numeric.vi	解一个带有常数微分方程的 $n$ 维齐次线性系统，结果为数值解
ODE 符号形式线性微分方程/ODE Linear System Symbolic.vi	解一个带有常数微分方程的 $n$ 维齐次线性系统，结果为符号解

下面通过举例说明这些函数在实际问题中的用法。

**例 11-6 常微分方程数值解举例。**

设河岸边点  $O$  的正对岸有一点  $A$ ，河宽  $OA=h$ ，两岸为水平直线，水流速度为  $a$ ，有一只鸭子要从点  $a$  游到点  $O$ ，设鸭子在静水中的游速为  $b$  ( $b>a$ )，且鸭子游动方向始终朝着点  $O$ ，求鸭子游过的轨迹方程。

解：设水流速度为  $a$  ( $|a|=a$ )，鸭子的游速为  $b$  ( $|b|=b$ )，则鸭子的实际运动速度为  $v=a+b$ 。取  $O$  点为坐标原点，河岸朝顺水方向为  $x$  轴， $y$  轴由对岸指向出发点，则鸭子的游动轨迹图如图 11.18 所示。

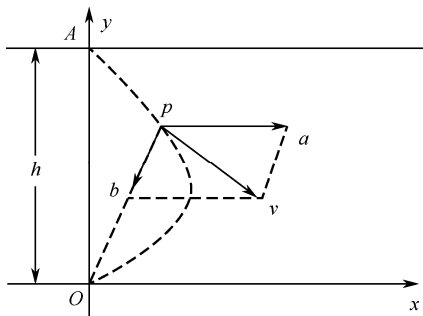


图 11.18 鸭子游过河的轨迹

设在时刻  $t$  鸭子位于点  $P(x, y)$ ，则鸭子的运动速度为：

$$\vec{v} = (v_x, v_y) = \left( \frac{dx}{dt}, \frac{dy}{dt} \right)$$

可以看出：

$$\begin{aligned}\vec{a} &= \{a, 0\} \\ \vec{b} &= -\frac{b}{\sqrt{x^2 + y^2}} \{x, y\}\end{aligned}$$

从而：

$$\vec{v} = \vec{a} + \vec{b} = \left( a - \frac{bx}{\sqrt{x^2 + y^2}}, -\frac{by}{\sqrt{x^2 + y^2}} \right)$$

由此得到微分方程：

$$\begin{cases} \frac{dx}{dt} = a - \frac{bx}{\sqrt{x^2 + y^2}} \\ \frac{dy}{dt} = -\frac{by}{\sqrt{x^2 + y^2}} \end{cases}$$

该微分方程可以用龙格-库塔方法解，因此这里使用 ODE 库塔四阶方法函数 (ODE Runge Kutta 4th Order.vi)。该函数的图标与端子如图 11.19 所示。

其中  $X$  表示求导的变量，time start 和 time end 代表  $t$  的范围， $h$  为求解的步长， $X0$  为各求导变量的初值，time 为  $t$  的变量符号， $F(X, t)$  为微分方程的右边部分函数。Times 为计算中对  $t$  求导的各点， $X$  Values 为求导的各变量的输出值。

用户只需要简单地将微分方程的各种参数作为该函数的输入就可以得到微分方程的数值

解。本例所得前面板和程序框图如图 11.20 所示。

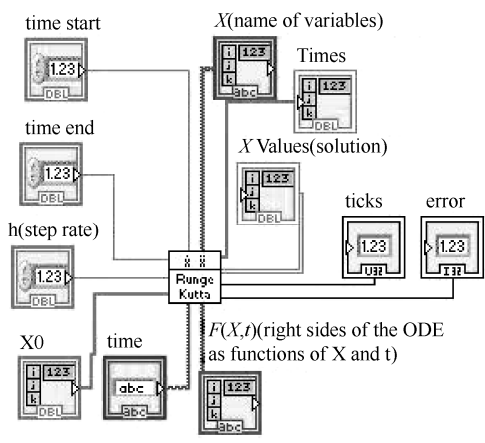


图 11.19 ODE 库塔四阶方法函数的图标及端子

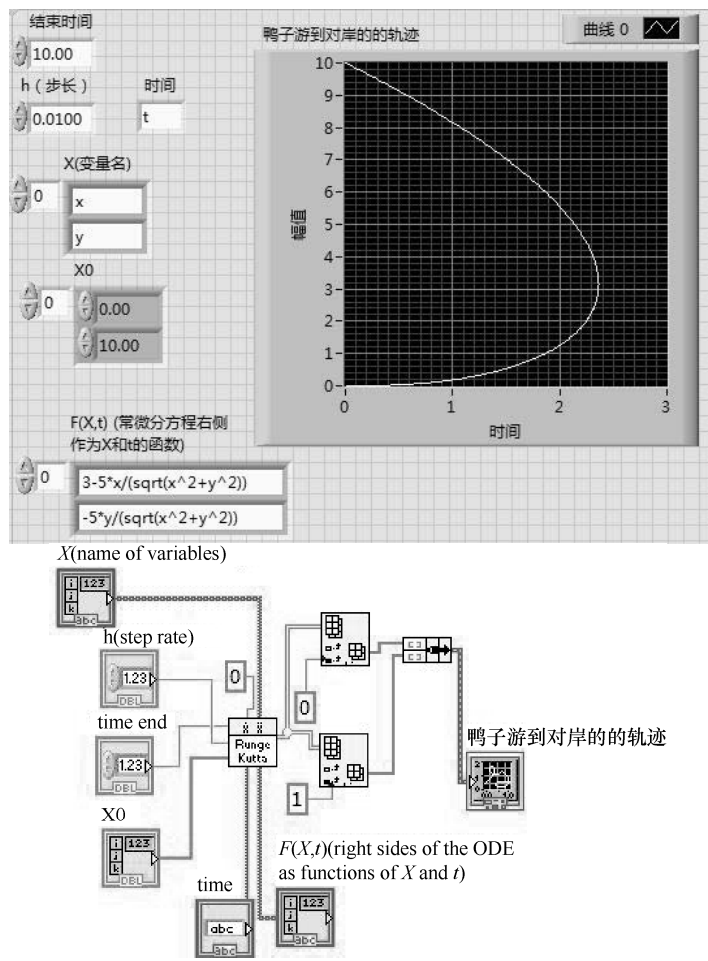


图 11.20 鸭子游到对岸的轨迹

## 11.9 几 何

在工程计算中，经常需要对空间几何进行坐标或角度变换。LabVIEW 提供了现成的函数用于几何坐标或角度变换，这些函数位于函数选板的“数学（Mathematics）→几何（Geometry）”面板下，如图 11.21 所示。

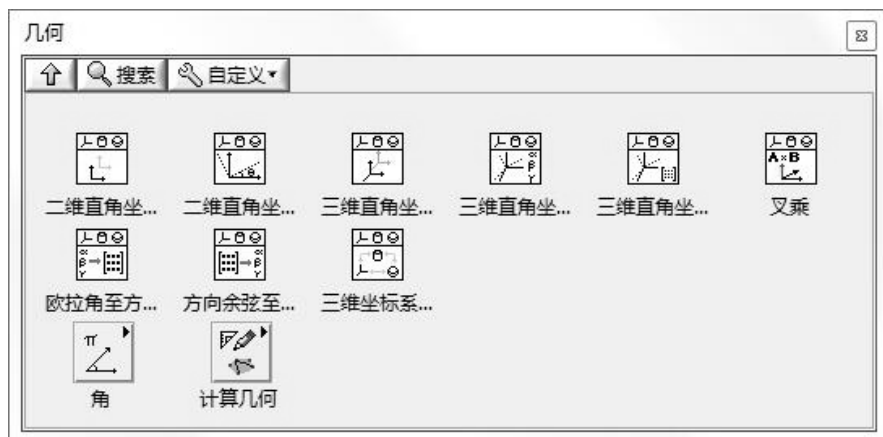


图 11.21 “几何”面板

这些函数的功能如表 11.3 所示。

表 11.3 几何函数功能列表

名 称	功 能
二维直角坐标系平移/2D Cartesian Coordinate Shift.vi	二维笛卡尔坐标平移，输入可以是多点也可以是单点
二维直角坐标系旋转/2D Cartesian Coordinate Rotation.vi	逆时针方向旋转二维笛卡尔坐标，输入可以是多点也可以是单点
三维直角坐标系平移/3D Cartesian Coordinate Shift.vi	三维笛卡尔坐标平移，输入可以是多点也可以是单点
三维直角坐标系旋转（欧拉角）/3D Cartesian Coordinate Rotation(Euler).vi	通过欧拉角方法逆时针旋转三维笛卡尔坐标，输入可以是多点也可以是单点
三维直角坐标系旋转（方向余弦）/3D Cartesian Coordinate Rotation(Direction).vi	通过方向余弦方法逆时针旋转三维笛卡尔坐标，输入可以是多点也可以是单点
欧拉角至方向余弦转换/Euler Angles To Direction Cosines.vi	将欧拉角转换为 3×3 的方向余弦矩阵
方向余弦至欧拉角转换/Direction Cosines To Euler Angles.vi	将 3×3 的方向余弦矩阵转换为欧拉角
三维坐标系变换/3D Coordinate Conversion.vi	直角坐标、球坐标和柱坐标之间的坐标变换，输入可以是多点也可以是单点

## 11.10 公式解析

前面讲述的内容均是数值计算，其中数学运算的公式是确定的，即在程序运行过程中数学运算的公式是不可以改变的。在很多情况下，我们还希望在程序运行过程中根据实际情况更改计算公式，这时候的输入可以是一个字符串公式。这就需要用到 LabVIEW 的公式解析函数，对输入的字符串进行解析计算。

对字符串公式进行解析计算的函数位于函数选板的“数学（Mathematics）→脚本与公式（Scripts & Formulas）→公式解析（Formula Parsing）”面板下，如图 11.22 所示。



图 11.22 “公式解析”面板

下面对公式解析面板中的三个函数进行详细说明。

### 1) 解析公式字符串 (Eval Formula String.vi)

解析公式字符串函数用于对不含变量的一维公式进行计算。例如，`formula` 可以是“ $\sin(\pi(1/2)) + 2 \times 2 - 1$ ”，运算结果是 4.00。`formula` 的语法与表达式节点语法相同，如图 11.23 所示。

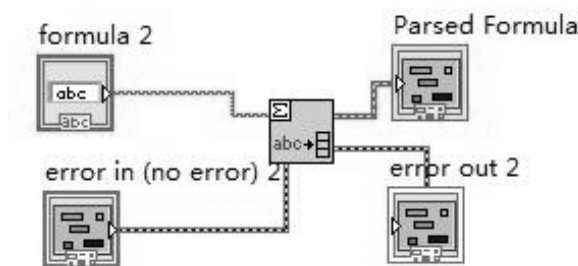


图 11.23 解析公式字符串

### 2) 解析公式节点 (Eval Formula Node.vi)

解析公式节点函数用于对多变量的多个公式同时进行计算，公式之间用分号隔开。

如图 11.24 所示。`Variables Input` 表示输入的自变量，`Input Values` 为相应的输入值，`Variables Output` 表示输出的变量，`Parsed Formula Node` 为对应的输出值。

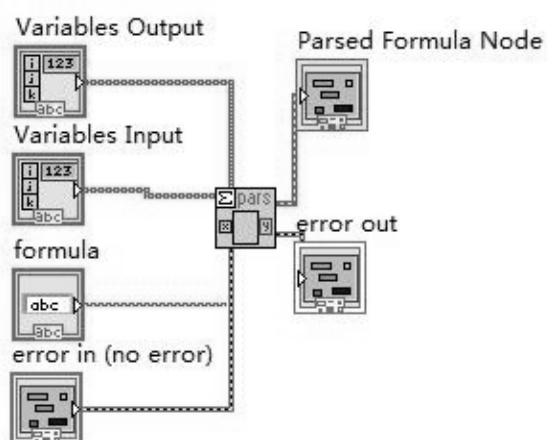


图 11.24 解析公式节点



## 第12章 波形分析

现实中数字信号无处不在。由于数字信号具有高保真、低噪声和便于处理的优点，所以得到了广泛的应用，例如，通信公司使用数字信号传输语音；广播、电视和高保真音响系统也都在逐渐数字化；太空中的卫星将测得的数据以数字信号的形式发送到地面接收站；对遥远星球和外部空间拍摄的照片也是采用数字方式处理，去除干扰、获得有用的信息；经济数据、人口普查结果和股票市场价格等都可以采用数字信号的形式获得。可用计算机处理的信号都是数字信号。

目前，对于实时分析系统，高速浮点运算和数字信号处理已经变得越来越重要。这些系统被广泛应用到生物医学数据处理、语音识别、数字音频和图像处理等各种领域。数据分析的重要性在于消除噪声干扰，纠正由于设备故障而遭到破坏的数据，或者补偿环境影响。

用于信号分析和处理的虚拟仪器执行的典型测量任务如下：

- 计算信号中存在的总谐波失真。
- 决定系统的脉冲响应或传递函数。
- 估计系统的动态响应参数，如上升时间、超调量等。
- 计算信号的幅频特性和相频特性。
- 估计信号中含有的交流成分和直流成分。

所有这些任务都要求在数据采集的基础上进行信号处理。

由采集得到的测量信号是等时间间隔的离散数据序列，LabVIEW 提供了专门描述它们的数据类型——波形数据。由它提取出所需要的测量信息，需要经过数据拟合来抑制噪声，减小测量误差，然后在频域或时域经过适当的处理才会得到所需要的结果。另外，一般来说，在构造这个测量波形时已经包含了后续处理的要求（如采样率的大小、样本数的多少等）。

LabVIEW 提供了大量的信号分析和处理函数。这些函数节点包含在函数选板中的信号处理子选板中，如图 12.1 所示。合理利用这些函数，会使测试任务达到事半功倍的效果。

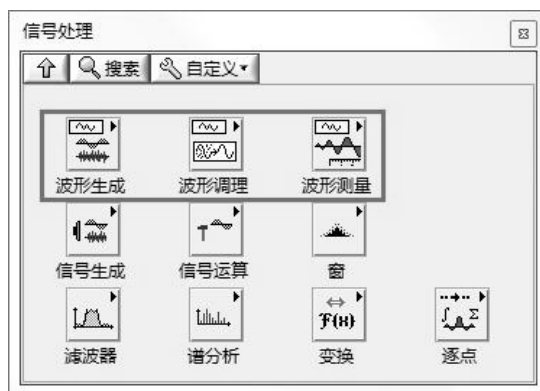


图 12.1 “信号处理”子选板

下面对信号的分析 and 处理中用到的函数节点进行介绍。

对于任何测试来说，信号的生成非常重要。例如，当现实世界中的真实信号很难得到时，可以用仿真信号对其进行模拟，向数/模转换器提供信号。常用的测试信号包括正弦波、方波、三角波、锯齿波和各种噪声信号以及由多种正弦波组成的多频信号。音频测试中最常见的是正弦波。正弦信号波形常用来判断系统的谐波失真度。合成正弦波信号广泛应用于测量互调失真或频率响应。

## 12.1 波形生成

LabVIEW 提供了大量的波形生成节点，它们位于“函数选板→信号处理→波形生成”子选板中，如图 12.2 所示。

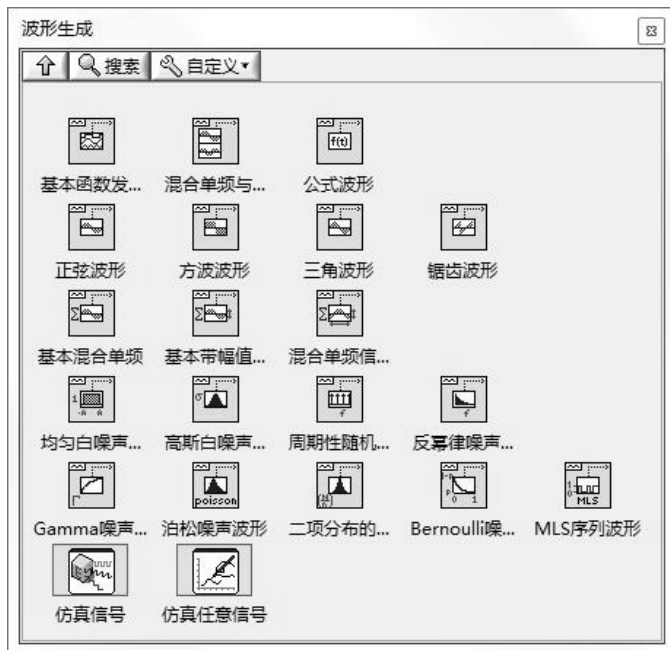


图 12.2 “波形生成”子选板

使用这些波形生成函数可以生成不同类型的波形信号和合成波形信号。下面对这些波形生成函数节点的图标及其使用方法进行介绍。

- 基本函数发生器：该项功能为产生并输出指定类型的波形。该 VI 会记住前一个波形的时间标识，并从前一个时间标识后面继续增加时间标识。它将根据信号类型、采样信息、占空比及频率的输入量来产生波形。利用基本函数发生器节点可以产生不同形式的信号波形，其频率、幅值和相位等参数可调。
- 公式波形：该项功能为生成公式字符串所规定的波形信号。
- 正弦波形：该项功能为产生正弦波形。该 VI 是重复输入的，因此可用于仿真连续采

集信号。如果重复信号输入端为 FALSE，接下来对 VI 的调用将产生下一个包含  $n$  个采样点的波形。如果重置信号输入端为 FALSE，该 VI 记忆当前 VI 的相位信息和时间标识，并据此来产生下一个波形的相关信息。利用正弦波形 VI 可以产生不同形式的信号波形，其频率、幅值和相位等参数可调。

- 基本混合单频：该项功能为产生多个正弦信号的叠加波形。所产生额定信号的频率在特定频率处是脉冲而在其他频率处是 0。根据频率和采样信息产生单频信号。单频信号的相位是随机的，它们的幅值相等，最后将这些单频信号进行生成。
- 混合单频与噪声波形：该项功能为产生一个包含正弦单频、噪声及直流分量的波形信号。
- 基本带幅值混合单频：该项功能为产生多个正弦信号的叠加波形。所产生的信号的频率谱在特定频率处是脉冲而其他频率处是 0。单频的个数由单频频率、单频幅值及单频相位端口输入数组的大小决定。使用单频频率、单频幅值、单频相位端口输入的信息将产生正弦单频，最后将所有产生的单频信号合成。
- 均匀白噪声波形：该项功能为产生伪随机白噪声。该 VI 对所产生的白噪声的相关参数进行调节，波形图中显示了所产生的白噪声波形信号。
- 周期性随机噪声波形：输出数组包含了一个整周期的所有频率。每个频率成分的幅度谱由幅度谱输入决定，且相位是随机的。输出的数组也可以认为是具有相同幅值随机相位的正弦信号的叠加和。
- Bernoulli 噪声波形：该项功能为产生伪随机 0-1 信号。信号输出的每一个元素经过取 1 概率的输入值运算。如果取 1 概率输入端的值为 0.7，那么信号输出的每一个元素将有 70% 的概率为 1，有 30% 的概率为 0。

## 12.2 波形调理

波形调理主要用于对信号进行数字滤波和加窗处理。波形调理 VI 节点位于“函数选板→信号处理→波形调理”子选板中，如图 12.3 所示。



图 12.3 “波形调理”子选板

下面对波形调理选板中包含的 VI 及其使用方法进行介绍。

- **数字 FIR 滤波器**：数字 FIR 滤波器可以对单波形和多波形进行滤波。如果对多波形进行滤波，则 VI 将对每一个波形进行相同的滤波。信号输入端和 FIR 滤波器规范输入端的数据类型决定了使用哪一个 VI 多态实例。该 VI 根据 FIR 滤波器规范和可选 FIR 滤波器规范的输入数组来对波形进行滤波。如果对多波形进行滤波，VI 将对每一个波形使用不同的滤波器，并且会保证每一个波形是相互分离的。
- **数字 IIR 滤波器**：输出信号通过数字 IIR 滤波器 VI 进行滤波。通过 IIR 滤波器规范和可选 IIR 滤波器规范簇中包含的输入控件可以对滤波器的滤波参数进行调节。当拓扑结构选择 Off 时，滤波器被关闭，波形图中输出的是仿真信号 VI 输出的波形信号。数字 IIR 滤波器可以对单个波形、多个波形中的信号进行滤波。
- **连续卷积 (FIR)**：该项功能将单个或多个信号和一个或多个具有状态信息的 kernel 相卷积，该节点可以连续调用。
- **按窗函数缩放**：该项功能对输入的时域信号加窗。根据信号输入的类型不同将使用不同的多态实例。
- **波形对齐 (连续)**：该项功能将波形按元素对齐，并返回对齐的波形。根据波形输入端输入的波形类型不同将用不同的多态 VI。
- **波形对齐 (单次)**：该项功能将波形按元素对齐，并返回对齐的波形。根据连线至波形输入端的数据类型可以确定使用的多态实例。
- **滤波器**：该 Express VI 用于通过滤波器和窗对信号进行处理。在“函数选板→Express→信号分析”子选板中也包含该 VI。滤波器 Express VI 也可以像其他 Express VI 一样对图标显示样式进行改变。当滤波器 Express VI 放置在程序框图上时，将弹出“配置滤波器”对话框。使用鼠标左键双击滤波器图标或者在右键快捷菜单中选择“属性”选项也会显示该对话框。在该对话框中可以对滤波器 Express VI 的参数进行设置。
- **对齐和重采样**：该 Express VI 用于改变开始时间，对齐信号或改变时间间隔，对信号进行重采样。该 Express VI 返回经调整的信号。该 Express VI 也可以像其他 Express VI 图标一样改变显示样式。
- **触发与门限**：触发与门限 Express VI 用于使用触发、提取信号中的一个片段。触发器组状态可基于开启或停止触发器的阈值，也可以是静态的。触发器为静态时，触发器立即启动，Express VI 返回预定数量的采样。该 Express VI 也可以像其他 Express VI 图标一样改变显示样式。

## 12.3 波形测量

使用波形测量选板中的 VI 进行最基本的时域和频域测量，如直流、平均值、单频频率/幅值/相位测量、谐波失真测量和信噪比及 FFT 测量等。波形测量 VI 在“函数选板→信号处理→波形测量”子选板中，如图 12.4 所示。

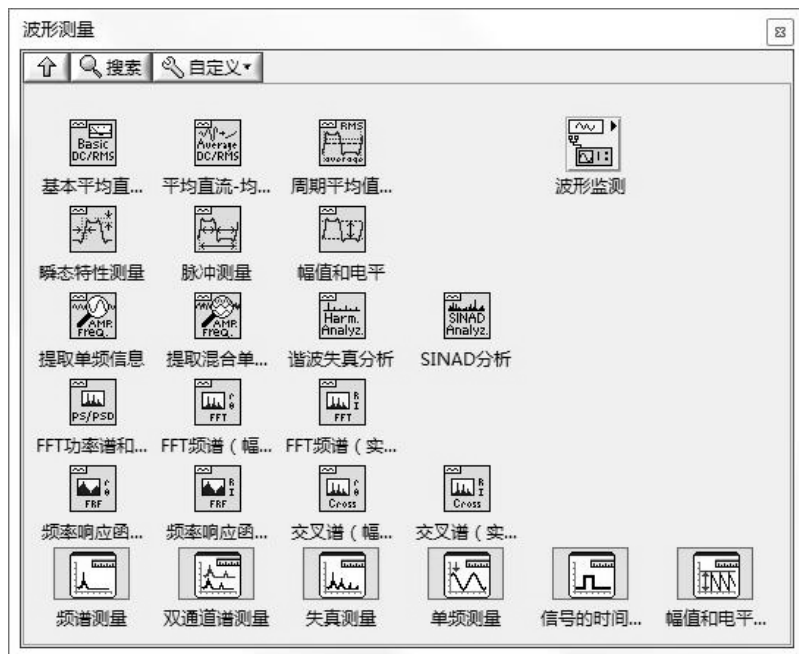


图 12.4 “波形测量”子选板

- 基本平均直流-均方根：该项功能从信号输入端输入一个波形或数组，对其加窗，根据平均类型输入端口的值计算加窗信号口信号的平均直流及均方根。信号输入端输入的信号类型不同，将使用不同的多态 VI 实例。
- 瞬态特性测量：该项功能输入一个波形或波形数组，测量其瞬态持续时间（上升时间或下降时间）、边沿斜率和前冲或过冲。信号输入端输入的信号类型不同，使用不同的多态 VI 实例。
- 提取单频信息：主要对输入信号进行检测，返回单频频率、幅值和相位信息。时间信号输入端输入的信号类型决定使用多态 VI 的实例。
- FFT 频谱（幅度-相位）：该项功能为计算时间信号的 FFT 频谱。FFT 频谱的返回结果是幅度和相位。时间信号输入端输入信号的类型决定使用何种多态 VI 实例。
- 频率响应函数（幅度-相位）：该项功能为计算输入信号的频率响应及相关性。结果返回幅值相位及相关性。一般来说时间信号 X 是激励，而时间信号 Y 是系统的响应。每一个时间信号对应一个单独的 FFT 模块，因此必须将每一个时间信号输入到一个 VI 中。
- 频谱测量：频谱测量 Express VI 用于进行基于 FFT 的频谱测量，如信号的平均幅度频谱、功率谱和相位谱。该 Express VI 也可以像其他 Express VI 图标一样改变显示样式。
- 失真测量：用于在信号上进行失真测量，如音频分析、总谐波失真（THD）、信号与噪声失真比（SINAD）。该 Express VI 也可以像其他 Express VI 图标一样改变显示样式。
- 幅值和电平测量：用于测量电平和电压。该 Express VI 也可以像其他 Express VI 图标一样改变显示样式。

## 12.4 信号分析

### 12.4.1 信号的时域分析

信号时域分析是指在时间域上对信号的时域参数进行测量和计算，从而提取有助于研究和分析的信号时域特性。时域分析往往是对原始信号进行分析的第一个步骤，信号的时域特性往往也是使用其他方法进行分析的重要参考和基础。

信号时域分析函数位于函数选板的“信号处理→信号运算（Signal Operation）”面板下，如图 12.5 所示。该函数面板提供的分析函数有直交流成分检测、卷积、逆卷积、相关分析、微分、积分、尖峰捕获、门限检测和过度分析等。

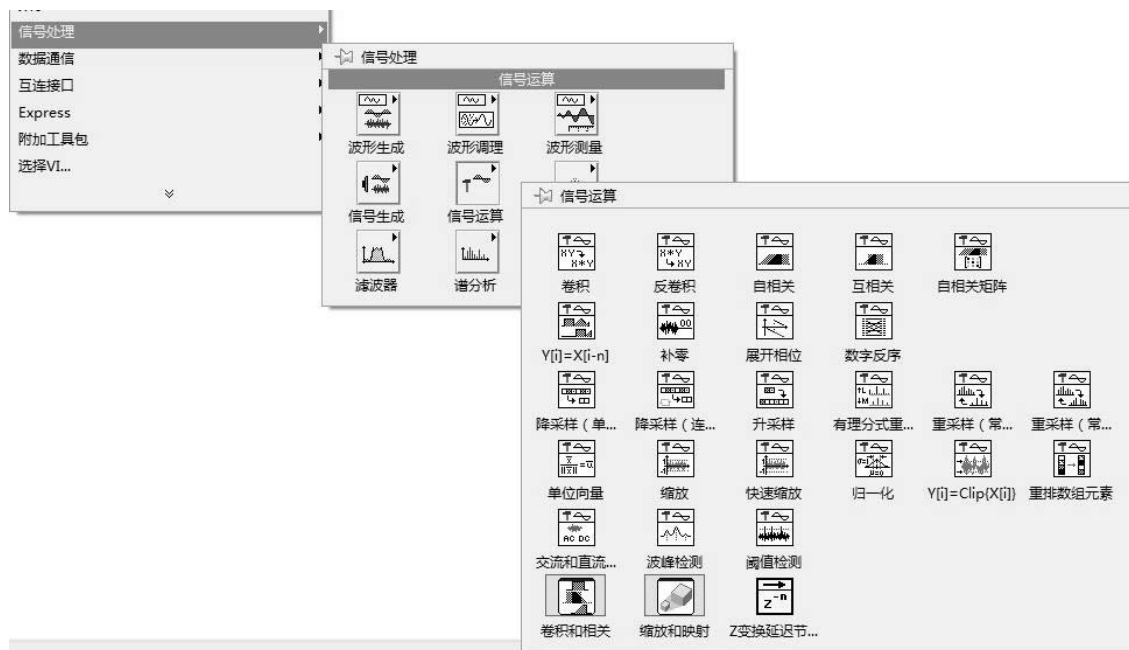


图 12.5 时域分析在函数选板中的位置

下面通过两个简单的例子来说明时域分析函数的应用。

**例 12-1** 采用 LabVIEW 语言实现多种基本波形显示，同时能够完成信号的直流值和均方根值的测量，并观察波形的变化情况。实现对输出波形的类型、幅值、频率、相位等参数进行改变的功能。

#### 1. 程序框图主要功能模块介绍。

本例主要利用“波形生成”和“波形测量”两个子选板中的“基本波形发生器”和“基本平均直流-均方值”函数模块实现。

“基本波形发生器”函数位于“函数”选板的“信号处理→波形测量”子选板中，如图 12.6 所示。此函数可以根据指定的信号类型，生成幅值、频率和初始相位自定义的基本波形。

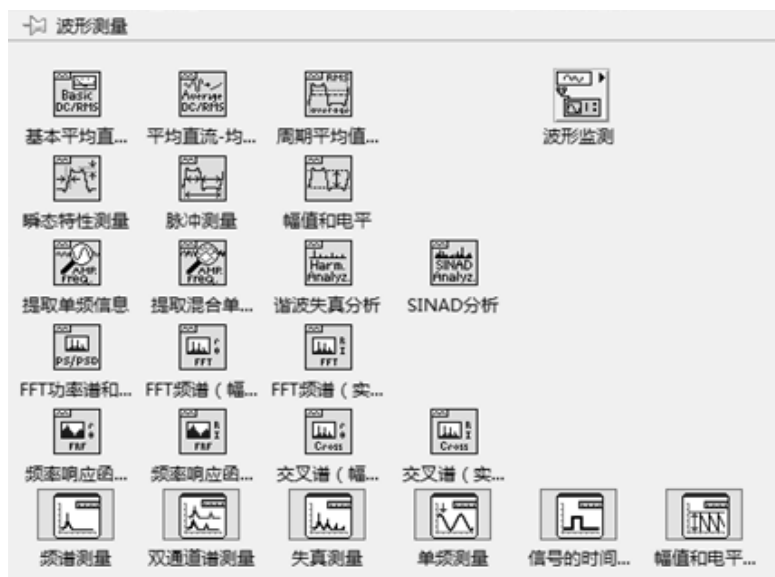


图 12.6 “波形测量”子选板

其图标及端口如图 12.7 所示，各端口的含义如下所述。

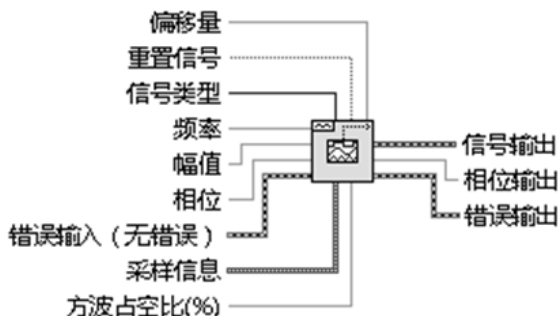


图 12.7 基本函数发生器图标及端口

**偏移量：**指定信号的直流偏移量，默认值为 0.0。

**重置信号：**如果是 TRUE，相位可重置为相位控件的值，时间标识可重置为 0，默认值为 FALSE。

**信号类型：**指要生成的波形类型，0（默认）为正弦波，1 为三角波，2 为方波，3 为锯齿波。

**频率：**指波形频率，单位为赫兹（Hz），默认值为 10。

**幅值：**指波形的幅值，也可以是峰值电压，默认值为 1.0。

**相位：**指波形的初始相位，当重置信号为 FALSE 时无效，默认值为 0。

**错误输入：**表明该节点运行前发生的错误条件。该输入提供标准错误输入。

**采样信息：**Fs 指每秒采样率，默认值为 1000；#s 指波形的采样点数，默认值为 1000。

**方波占空比：**表示方波在一个周期内高电平所占时间的百分比，仅当“信号类型”是方波时才有效，默认值为 50。

信号输出：指生成的波形。

相位输出：指波形的相位，单位是度（°）。

错误输出：包含错误信息，该输出提供标准错误输出。

“基本平均直流-均方根”函数位于“函数”选板的“信号处理→波形测量”子选板中，该函数模块用于计算输入波形或波形数组的直流值（DC）和均方根值（即有效值 RMS），其图标及端口如图 12.8 所示，端口含义如下所述。

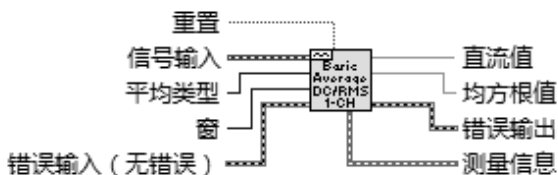


图 12.8 基本平均直流-均方根函数的图标及端口

**平均类型：**测量时使用的平均类型。VI 计算每个输入波形的 DC 和 RMS 值，因此平均时间由输入记录的长度确定。如“平均类型”为 Exponential，则 VI 通过对上一个 DC 和 RMS 值。其值为 0 表示 Linear，平均时间与记录长度相同；值为 1 表示 Exponential，时间常量为记录长度的一半。

**窗：**在 DC/RMS 计算前对记录应用的窗。如“平均类型”为 Exponential，LabVIEW 可忽略该输入。窗的类型有无窗、汉宁窗、Low side lobe 窗。

## 2. 设计/实现步骤。

### 1) 前面板设计

(1) 创建新 VI，命名为 DC-RMS.vi。其操作路径为“文件→新建 VI”。

(2) 放置数值控件、枚举控件及图形控件。

a. 执行“控件→新式→数值→数值输入控件”操作，添加 5 个数值输入控件并分别命名为“幅值”（默认值设置为 1.00）、“频率”（默认值设置为 5.00）、“相位”（默认值设置为 0.00）、“占空比（%）”（默认值设置为 50.00）和标准偏差（默认值设置为 0.1）。

b. 执行“控件→新式→数值→数值显示控件”操作，添加 2 个数值显示控件，分别命名为“直流量”和“均方根值”。

c. 执行“控件→新式→下拉列表和枚举→枚举”操作，添加 1 个枚举控件并且命名为“信号类型”，编辑项基于索引 0 依次为 Sine Wave, Triangle Wave, Square Wave, Sawtooth Wave，如图 12.9 所示。

d. 执行“控件→新式→布尔→停止按钮”操作，添加一个“停止”按钮，用于中断运行中的 VI 程序。

e. 放置波形图控件并命名为“WaveForm”，其操作路径是“控件→新式→图形→波形图”。





图 12.9 枚举类属性设置

前面板界面设计如图 12.10 所示。

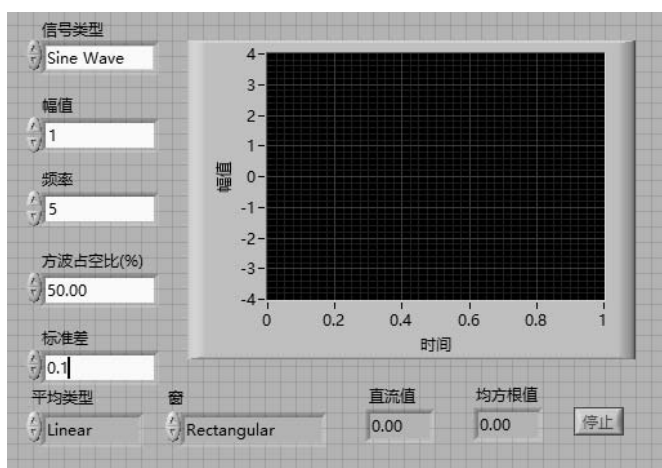


图 12.10 前面板设计

### 3. 程序框图的编辑

(1) 打开程序框图编辑窗口，相应的控件图标已经显示出来，其操作路径为“窗口→显示程序框图”。

(2) 放置 While 循环、基本函数发生器、基本平均直流-均方根等节点图标。

- a. 执行“函数→编程→结构→While 循环”操作，将所有节点拖入 While 循环中。
- b. 执行“函数→信号处理→波形生成→基本函数发生器”操作，添加基本函数发生

器节点。

c. 执行“函数→信号处理→波形生成→高斯白噪声波形”操作，添加高斯白噪声波形节点。

d. 执行“函数→信号处理→波形测量→基本平均直流-均方根”操作，添加基本平均直流-均方根节点。

(3) 将“基本波形发生器”与“高斯白噪声波形”节点相加后与“基本平均直流-均方根”函数节点的端子与参数输入连接起来，输出平均直流和均方根值。将“停止”按钮的节点和 While 循环的条件终端连接起来。

设计完毕后，测量信号平均直流-均方根值实例的程序框图如图 12.11 所示。

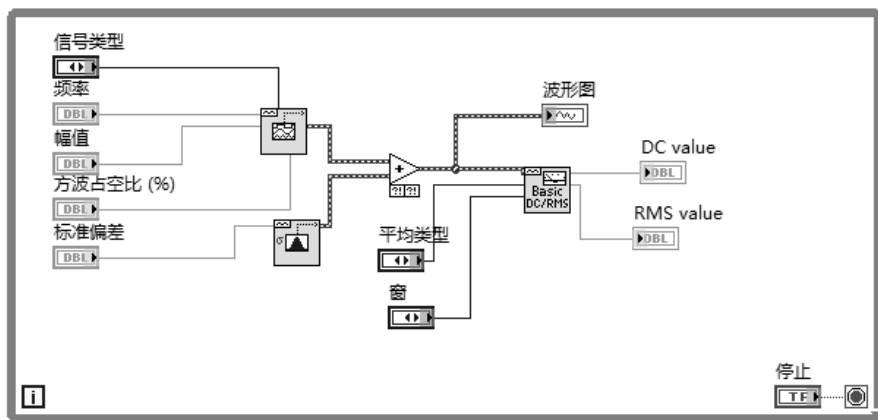


图 12.11 平均直流-均方根程序框图

#### 4. 运行程序

单击运行按钮，可在波形图控件中观察到“signal type”是“Sine Wave”运行结果，下方的显示控件给出了输入参数信息，如图 12.12 所示单击终止执行按钮或“停止”按钮，程序将停止运行。

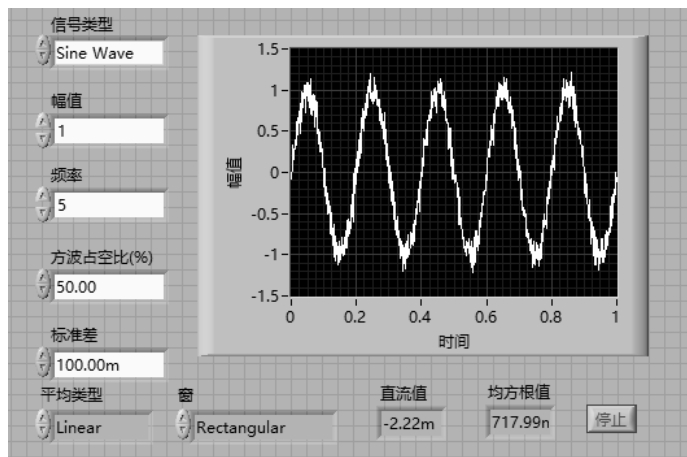


图 12.12 程序运行图

### 例 10-2 相关分析及其 LabVIEW 实现。

相关是两个变量之间的线性依存关系。相关分析在信号分析处理中有着广泛的应用，如信号的时延估计、周期成分的检测、相关滤波等。自相关函数的一个重要应用是检测信号中的周期成分。互相关函数可以测量系统的延时，提取噪声中的有用信息（相关滤波）。LabVIEW 提供了自相关和互相关两个 VI，位于函数选板中“信号处理→信号运算”子选板，如图 12.13 所示。

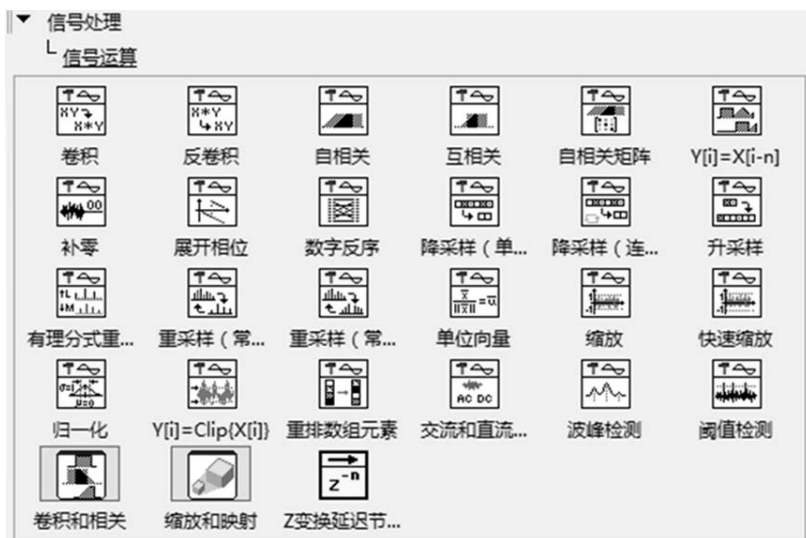


图 12.13 信号运算子选板

自相关 VI 用来计算输入序列  $X$  的自相关，自相关 VI 图标及端口如图 12.14 所示。

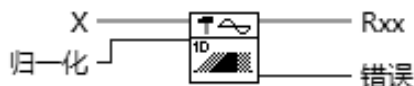


图 12.14 自相关 VI 图标及端口

$X$ ：是输入序列。

归一化：指定用于计算  $X$  自相关的归一化方法。有三种方法可供选择：none（默认），unbiased，biased。

$R_{xx}$ ：是  $X$  的自相关。

错误：返回 VI 的任何错误或警告。将错误连接至错误代码和错误簇转换 VI，可将错误代码或警告转换为错误簇。

互相关用来计算输入序列  $X$  和  $Y$  的互相关，其图标及端口如图 12.15 所示。

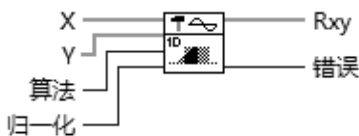


图 12.15 互相关 VI 图标及端口

X: 输入的实数或复数值序列;

Y: 输入的实数或复数值序列;

算法: 指定使用的相关方法。值为 **direct** 时, VI 使用线性卷积的 **direct** 方法计算互相关; 值为 **frequency domain** 时, VI 使用基于 FFT 的方法计算互相关。如 X 和 Y 较小, **direct** 方法通常更快。如 X 和 Y 较大, **frequency domain** 方法通常更快。此外, 两个方法数值上存在微小的差异。

归一化: 用于指定计算 X 和 Y 互相关的归一化方法。有三种选择方法: **none** (默认), **unbiased**, **biased**。

Rxy: 输出的互相关序列。

错误: 返回 VI 中相应的任一种错误和警告。

### 1. 设计目的

本例基于 LabVIEW 2013 开发平台, 利用“信号生成”子选板中的“正弦信号”函数和“信号运算”子选板中的“互相关”函数等函数节点, 设计一个能够实现互相关功能的实例。

## 2. 设计步骤

### 1) 前面板的设计

(1) 创建一个新 VI, 操作路径为“文件→新建 VI”。

(2) 执行“控件-新式-数值-数值输入控件”操作, 添加 4 个输入控件, 依次命名为“频率 1”、“相位 1”、“频率 2”和“相位 2”。

(3) 执行“控件-新式-图形-波形图”操作, 添加 2 个波形显示控件, 分别命名为“信号波形”和“相关后的波形”。

最后完成前面板的设计, 如图 12.16 所示。

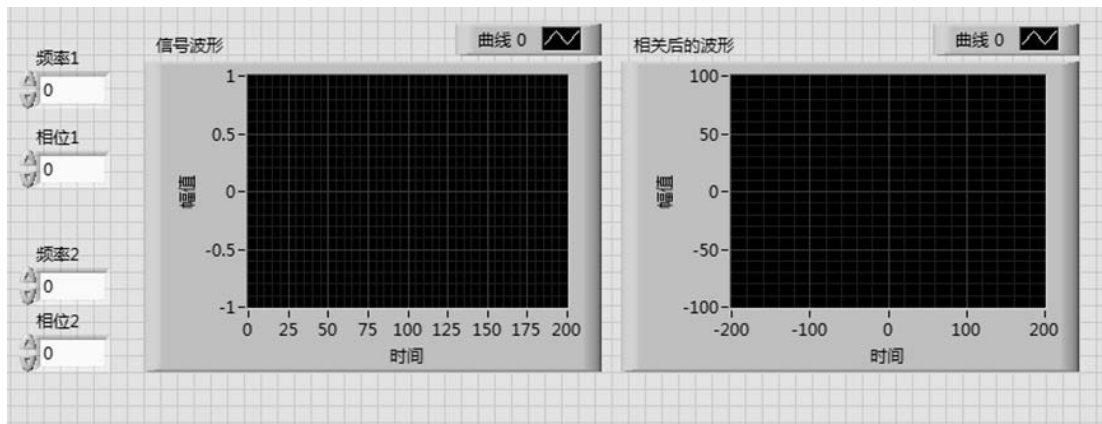


图 12.16 互相关前面板

### 2) 程序框图的设计

(1) 打开程序框图面板, 已经出现前面板所选择控件的图标, 再添加其他的正弦信号、互相关、创建数组、捆绑等节点。

- 执行“函数-信号处理-波形生成-正弦信号”操作，添加 2 个正弦函数节点。
- 执行“函数-信号处理-信号运算-互相关”操作，添加互相关函数节点。
- 执行“函数-新式-数组-创建数组”操作，添加创建数组节点。
- 执行“函数-新式-簇、类与变体-捆绑”操作，添加捆绑节点。
- 执行“函数-新式-数值-取负数、减 1”等操作。

(2) 分别将频率 1 和相位 1 与第一个正弦信号对应的端口相连接，将频率 2 和相位 2 与第二个正弦信号对应的端口相连接，正弦信号采样点数设置为 200。

(3) 将两个正弦信号输出通过创建数组节点后与捆绑节点相连，得到信号时域波形。

(4) 将两个正弦信号进行互相关后与捆绑节点相连，得到相关后的波形。

最后得到如图 12.17 所示的程序框图。

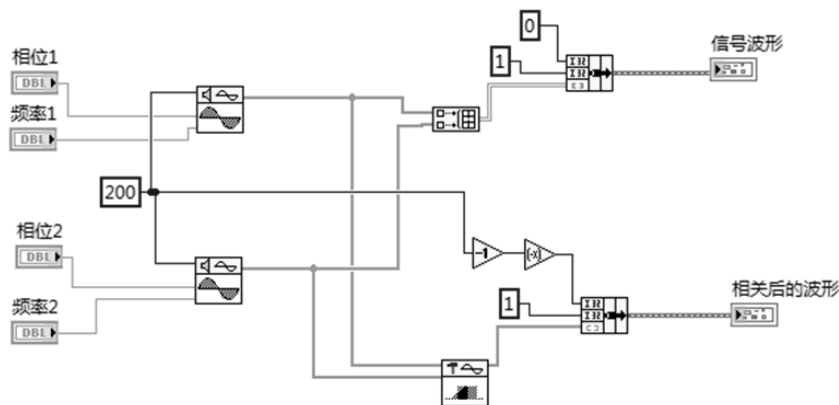




图 12.17 互相关程序框图

### 3) 运行结果

单击运行按钮 ，得到如图 12.18 所示的结果，可以看到两个频率相同，相位相差  $90^\circ$  的正弦波形，以及这两个信号互相关后的结果，单击终止执行按钮  将停止运行。

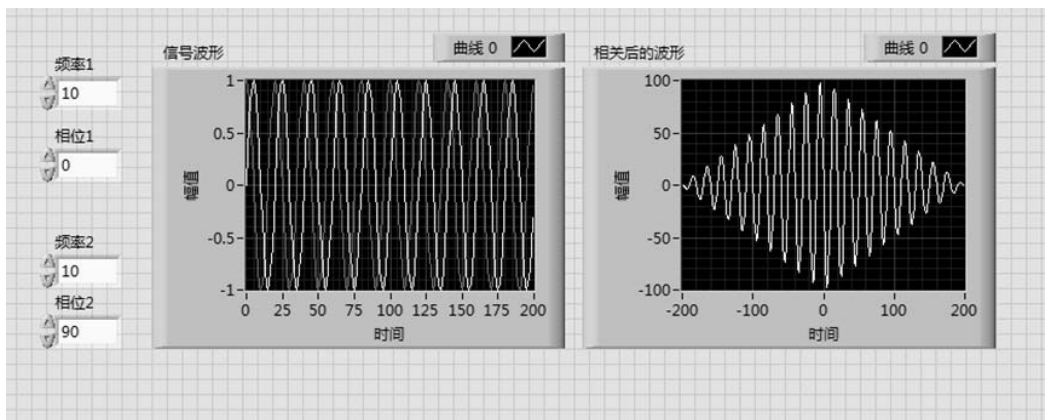
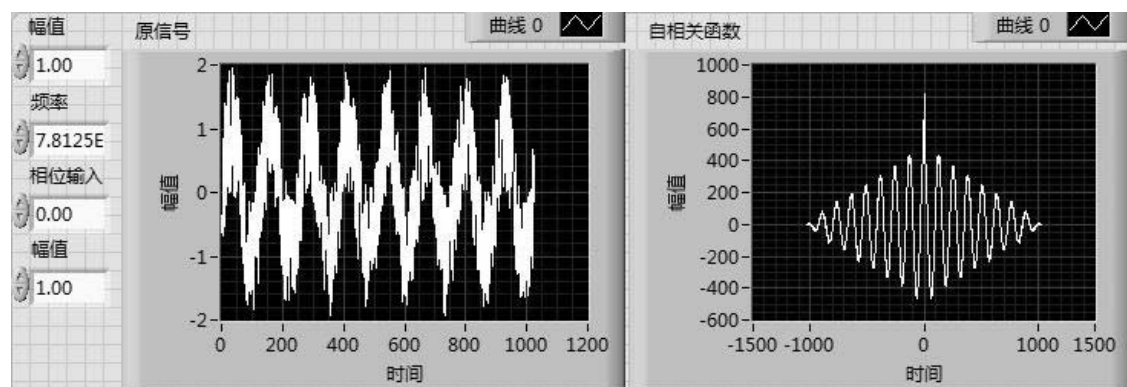


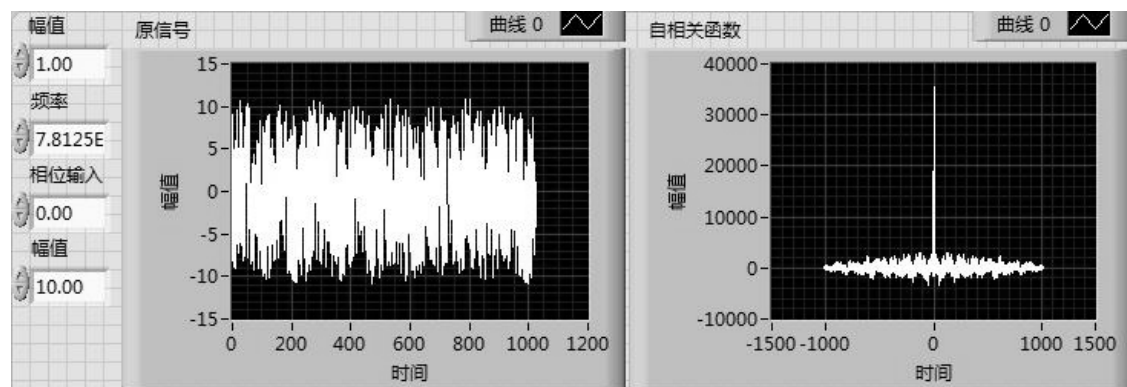
图 12.18 程序运行结果

## 例 12-2 周期信号检测。

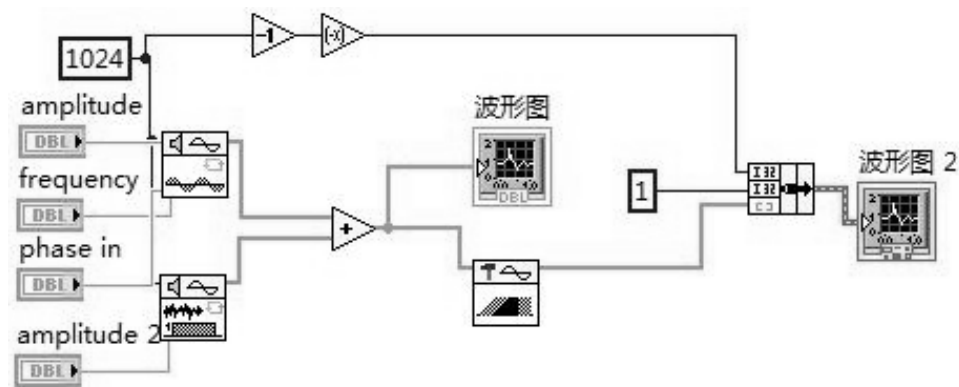
自相关函数常被用来检测信号中是否含有周期成分。如果信号中含有周期成分，则其自相关函数衰减很慢且具有明显的周期性，如图 12.19 所示。设原信号为一正弦波与噪声信号叠加而成，当噪声幅值与正弦波幅值相当时，可以看到自相关函数衰减很慢而且具有明显的周期性，如图 12.19 (a) 所示。如果增大噪声幅度，使其远大于正弦波幅度，从自相关函数图中就很难看到周期成分了，因为正弦周期信号已经被噪声淹没了，如图 12.19 (b) 所示。



a



b



c

图 12.19 自相关函数举例

## 12.4.2 信号的频域分析

频域分析函数被划分为两个面板：一个是“变换（Transforms）”面板，该面板实现的函数功能主要有傅里叶变换、Hilbert 变换、拉普拉斯变换等；另一个是“谱分析（Spectral Analysis）”面板，该面板所包含的函数主要包括功率谱分析、联合时域分析等。这两个函数面板均位于函数选板的信号处理面板下，如图 12.20 所示。



图 12.20 变换面板和谱分析面板

频域分析是数字信号处理中最常用、最重要的方法。下面通过两个例子来举例说明如何通过 LabVIEW 提供的 VI 函数实现信号的频域分析。

### 例 12-3 信号的傅里叶变换。

傅里叶变换是数字信号处理中重要的一个变换之一，它的意义在于使人们能够在频域中观察一个信号的特征。它的一个基本应用就是计算信号的频谱，通过频谱可以方便地观察和分析信号的频率组成成分。

现在通过 3 个正弦信号发生函数产生 3 个不同频率、不同振型的正弦信号，并将其叠加为同一个信号作为傅里叶变换函数的输入，如图 12.21 所示。

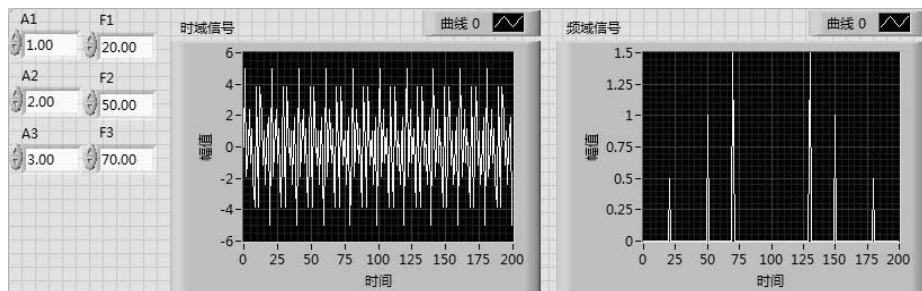


图 12.21 傅里叶变换

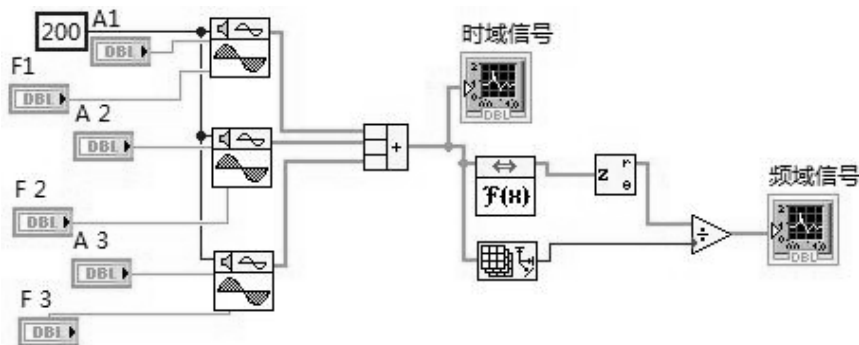


图 12.21 傅里叶变换 (续)

从图 12.21 可以看出, 傅里叶频谱中除了原有频率, 在  $\text{samples-f}$  的位置也有相应频率成分。这是由于 FFT.vi 函数计算得到的结果是采样信号频谱在采样区间  $[0, f_s]$  上的一段 ( $f_s$  为采样频率), 它不仅含有正频率成分, 而且还包含负频率成分。因此信号频率等于 20Hz 时, 180Hz 处出现的频谱实际上对应的频率为 -20Hz。如果不断地增大  $f$ , 可以看到正负频率对应的频谱逐渐靠近, 当  $f > f_s/2$  时就是采样定理所限制的结果, 因此为了能够获得正确的频谱, 采样时必须满足采样定理, 即  $f < f_s/2$ 。

**例 12-4** 利用选板中的“正弦波”和“FFT”函数, 实现对由两个不同频率正弦波生成的复杂周期信号的单边傅里叶变换。

一维实数、复数的 FFT 的 VI 图标及端口如图 12.22 所示。

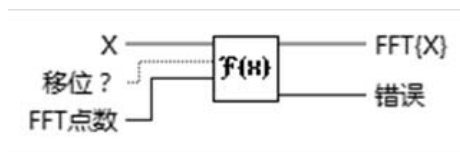


图 12.22 FFT 的 VI 图标及端口

其输入/输出参数说明如下。

**X:** 输入的实数或复数序列。

**移位?:** 指定 DC 元素是否位于 FFT {X} 中心, 默认值为 FALSE。

**FFT 点数:** FFT 点数是要进行 FFT 的长度。如 FFT 点数大于 X 的元素数, VI 将在 X 的末尾添加 0, 以匹配 FFT 点数的大小。如 FFT 点数小于 X 的元素数, VI 只使用 X 中的前  $n$  个元素进行 FFT,  $n$  是 FFT 点数。如 FFT 点数小于等于 0, VI 将使用 X 的长度作为 FFT 点数。

**FFT {X}:** 是 X 的 FFT。

**错误:** 返回 VI 的任何错误或警告。将错误连接至错误代码和错误簇转换 VI, 可将错误代码或警告转换为错误簇。

### 1. 设计步骤:

#### 1) 前面板的设计

(1) 创建新 VI, 操作路径为“文件→新建 VI”。



(2) 放置数值控件及图形控件。

a. 执行“控件→新式→数值输入控件”操作，添加 6 个输入控件并分别命名为“频率 1 (Hz)”（默认值设置为 5.0000），“幅值 1”（默认值设置为 1.00），“频率 2 (Hz)”（默认值设置为 10.0000），“幅值 2”（默认值设置为 1.00），“采样频率”（默认值为 100），“采样点数”（默认值为 100）。

b. 执行“控件→新式→图形→波形图”操作，添加两个波形图控件，分别命名为“信号波形”、“单边 FFT 变换”。

最后得到如图 12.23 所示的前面板的界面。

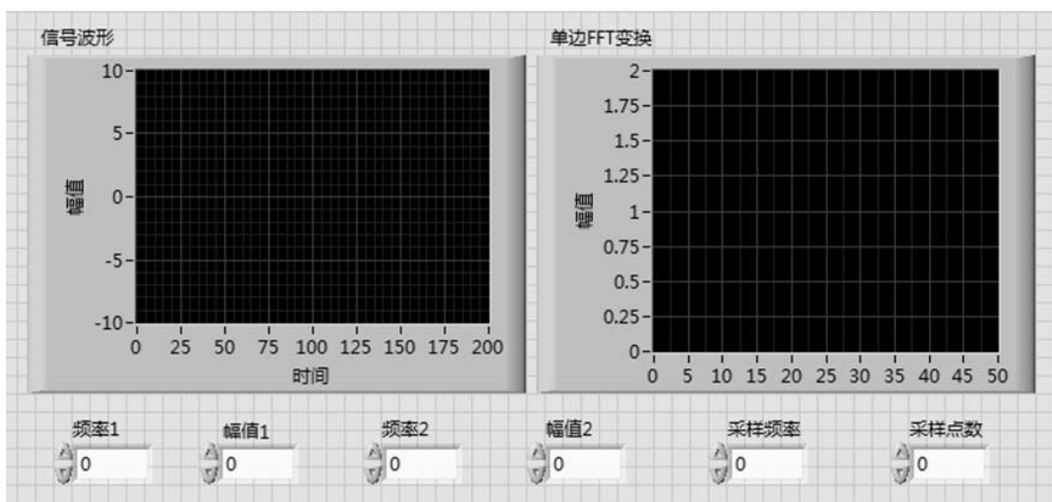


图 12.23 单边傅里叶变换前面板

## 2) 程序框图的设计

(1) 执行“窗口→显示程序框图”切换到程序框图编辑窗口。

(2) 放置正弦波、FFT、捆绑簇、数组大小、数组子集、复数至极坐标转换、除法、加法等函数节点。具体操作流程如下所述。

a. 添加两个“正弦波”函数，路径：函数→编程→波形→模拟波形→波形生成→正弦波。“正弦波”函数使用的输入参数“频率”是归一化频率类型，其单位为 cycles/sample，通过信号频率与采样频率相除完成。采样点数、幅值、频率和采样频率做除法后分别连接到正弦波函数输入端子。通过“加法”节点，实现两正弦波的相加。

b. 添加“FFT”函数，路径：函数→编程→信号处理→变换→FFT。将两个正弦波相加后的信号与 FFT 函数连接起来。

c. 添加“数组大小”和“数组子集”函数，路径：函数→编程→数组→数组大小、数组子集，实现取出 FFT 变换输出数组的一半。

d. 添加“复数至极坐标转换”函数，路径：函数→编程→数值→复数→复数至极坐标转换，端子 Z 为复数形式， $r$  为极径， $e$  为极角，实现把 FFT 变换得到的复数组分解为极坐标得到振幅，之后乘以 2 将幅值扩大一倍。

e. 添加“捆绑”函数，路径：函数→编程→簇、类族与变体→捆绑；采样频率和采样点

数做除法后与常量值 0、极坐标数据进行捆绑，然后输出单边 FFT 变换。

最后得到如图 12.24 的程序框图。

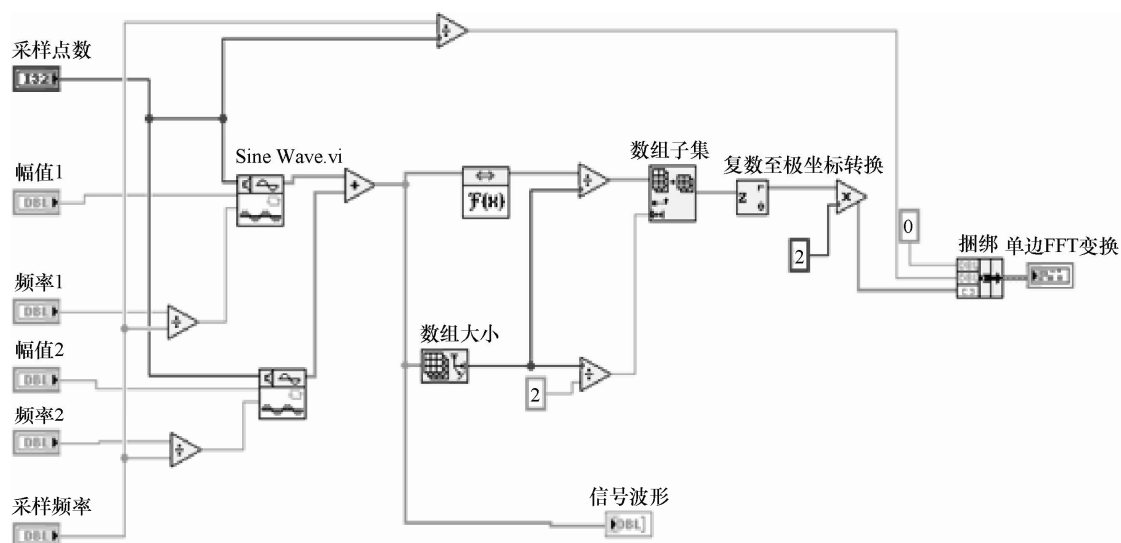




图 12.24 单边傅里叶变换程序框图

### 3) 执行结果

单击运行按钮 ，结果如图 12.25 所示，从频谱图中可观察到两频率混合的正弦波经过傅里叶变换后，完成了从时域到频域的映射，频谱图上的频率成分包含 5Hz 和 10Hz，与前面板上设置的频率参数相吻合。单击终止执行按钮  将停止运行。

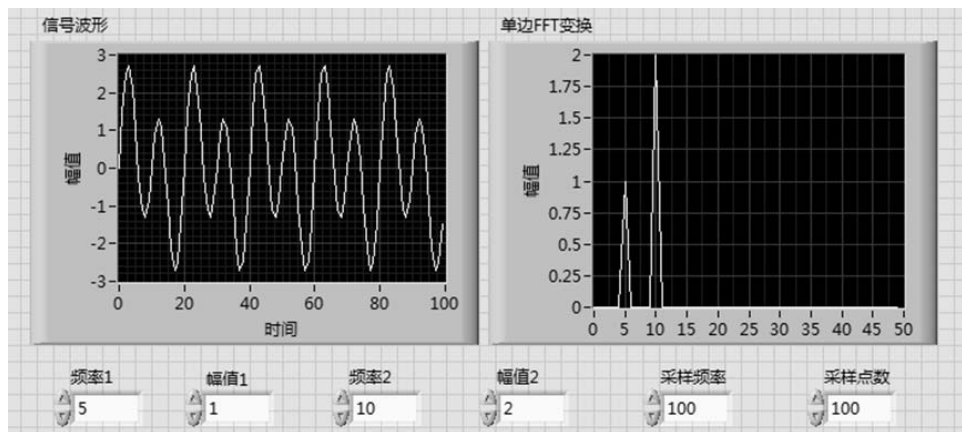


图 12.25 运行结果

### 例 12-5 功率谱分析及 LabVIEW 实现。

信号的功率谱分析是信号频域分析的一种常用方法。功率谱是指用密度的概念来表示信号功率在各频率点的分布情况。LabVIEW 提供了许多用于功率谱分析与计算的 VI，如自功率谱、互功率谱、非均匀采样数据的功率谱等，它们位于函数选板中“信号处理→谱分析”子选板，如图 12.26 所示。



图 12.26 谱分析子选板

功率谱 VI 的图标及端口如图 12.27 所示，其端口的含义如下所述。

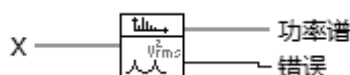


图 12.27 功率谱 VI 图标及其端口

**X:** 是输入序列。

**功率谱:** 返回 **X** 的双边功率谱。如输入信号以伏特 (V) 为单位，功率谱的单位为伏特的均方根 (rms) 的平方 ( $V_{rms}^2$ )。如输入信号不是以伏特为单位，则功率谱的单位为输入信号单位的均方根的平方。

**错误:** 返回 VI 的任何错误或警告。将错误连接至错误代码和错误簇转换 VI，可将错误代码或警告转换为错误簇。

自功率谱 VI 的图标及端口如图 12.28 所示，其端口的含义如下所述。

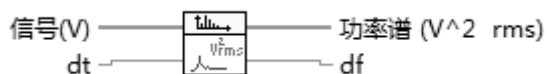


图 12.28 自功率谱 VI 图标及其端口

**信号:** 指定输入的时域信号，通常以伏特为单位。时域信号必须包含至少三个周期的信号才能进行有效的估计。

**dt:** 是时域信号的采样周期，通常以秒为单位。设置 **dt** 为  $1/f_s$ ， $f_s$  是时域信号的采样频率，默认值为 1。

**功率谱:** 返回单边功率谱。如输入信号以伏特 (V) 为单位，功率谱的单位为伏特的均方根的平方 ( $V_{rms}^2$ )。如输入信号不是以伏特为单位，则功率谱的单位为输入信号单位的均方根的平方。

**Df:** 如 **dt** 以秒为单位，该值是功率谱的频率间隔，以赫兹为单位。

互功率谱 VI 的图标及端口如图 12.29 所示，其端口的含义如下所述。

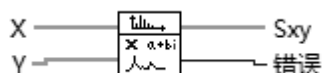


图 12.29 互功率谱 VI 图标及其端口

X: 是第一个输入序列。

Y: 是第二个输入序列。

Sxy: 是输入信号 X 和 Y 的单边互功率谱。

错误: 返回 VI 的任何错误或警告。将错误连接至错误代码和错误簇转换 VI, 可将错误代码或警告转换为错误簇。

幅度谱和相位谱 VI 的图标及端口如图 12.30 所示, 其端口的含义如下所述。



图 12.30 幅度谱和相位谱 VI 图标及其端口

信号: 指定输入的时域信号, 通常以伏特为单位。时域信号必须包含至少三个周期的信号才能进行有效的估计。

展开相位: 当值为 TRUE 时, 对输出幅度谱相位启用展开相位, 默认值为 TRUE。如展开相位的值为 FALSE, VI 不展开输出相位。

dt: 表示时域信号的采样周期, 通常以秒为单位。设置 dt 为  $1/f_s$ ,  $f_s$  是时域信号的采样频率, 默认值为 1。

幅度谱大小: 返回单边功率谱的幅度。如输入信号以伏特 (V) 为单位, 幅度谱大小的单位为伏特的均方根 ( $V_{rms}$ )。如输入信号不是以伏特为单位, 则幅度谱大小的单位为输入信号单位的均方根。

幅度谱相位: 指单边幅度谱相位, 以弧度为单位。

df: 如 dt 以秒为单位, 该值是功率谱的频率间隔, 以赫兹为单位。

## 1. 设计目的

本例利用“波形生成”子选板中的“正弦函数”函数和“谱分析”子选板中的“互功率谱”函数, 设计一互功率谱测量虚拟仪器, 将时域信号经计算处理后的信号功率谱显示出来, 并且可以实现波形参数信息的设置和显示。

## 2. 设计步骤

### 1) 前面板的设计

(1) 创建新 VI, 操作路径为“文件→新建 VI”。

(2) 放置数值控件、布尔控件及图形控件。

a. 执行“控件→新式→数值输入控件”操作, 添加 8 个输入控件并分别命名为“幅值 A (默认值设置为 1.00)”, “频率 A (默认值设置为 5.00)”, “相位 A (默认值为 0)”, “偏移量 A (默认值为 90)”, “幅值 B (默认值设置为 1.00)”, “频率 B (默认值设置为 10.00)”, “相位 A (默认值为 90)”, “偏移量 A (默认值为 0)”。

b. 执行“控件→新式→布尔→停止按钮”操作, 添加一个“停止”按钮, 用于停止运行中的 VI 程序。

c. 执行“控件→新式→图形→波形图”操作，添加两个波形图控件，分别命名为“时域波形”、“互功率谱”。

最后得到如图 12.31 所示前面板的界面。

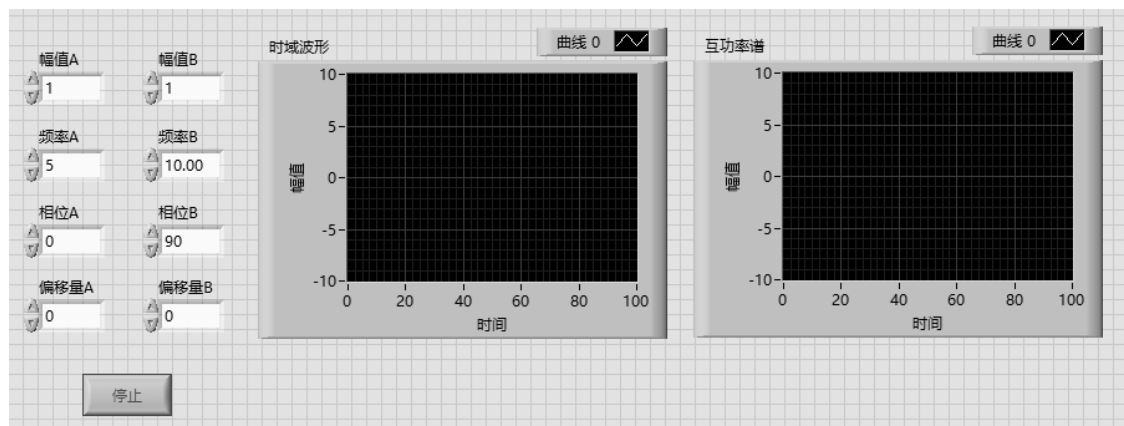


图 12.31 互功率谱前面板

## 2) 功率谱程序框图的设计

(1) 执行：“窗口→显示程序框图”，切换到程序框图编辑窗口，相应的控件图标已经显示出来。

(2) 放置 While 循环、正弦波、创建数组、互功率谱、等待 (ms) 等节点。具体操作流程如下：

- a. 执行“函数→信号处理→波形生成→正弦波形”操作，添加正弦波形节点。
- b. “函数→编程→数组→创建数组操作”，添加创建数组节点。
- c. “函数→编程→定时→等待 (ms) 操作”，添加等待 (ms) 节点。
- d. 执行“函数→编程→结构→While 循环”操作，将所有节点拖入 While 循环中。

(3) 将幅值 A、频率 A、相位 A、偏移量 A 与第一个正弦波形节点相应端口连接，幅值 B、频率 B、相位 B、偏移量 B 与第二个正弦波形节点相应端口连接。

(4) 将两个正弦波形的输出与创建数组函数的输入端口相连接，并将创建数组函数的输出与时域波形节点端口相连。



(5) 将两个正弦波形的输出与互功率谱函数的输入端口相连接，并将互功率谱函数的输出与互功率谱波形节点端口相连。

(6) 将“停止”按钮的节点和 While 循环的条件终端连接起来。

(7) 等待时间设置为 100ms。

设计的最后程序框图如图 12.32 所示。

## 3) 运行结果

单击运行按钮，结果如图 12.33 所示，单击终止执行按钮将停止运行。

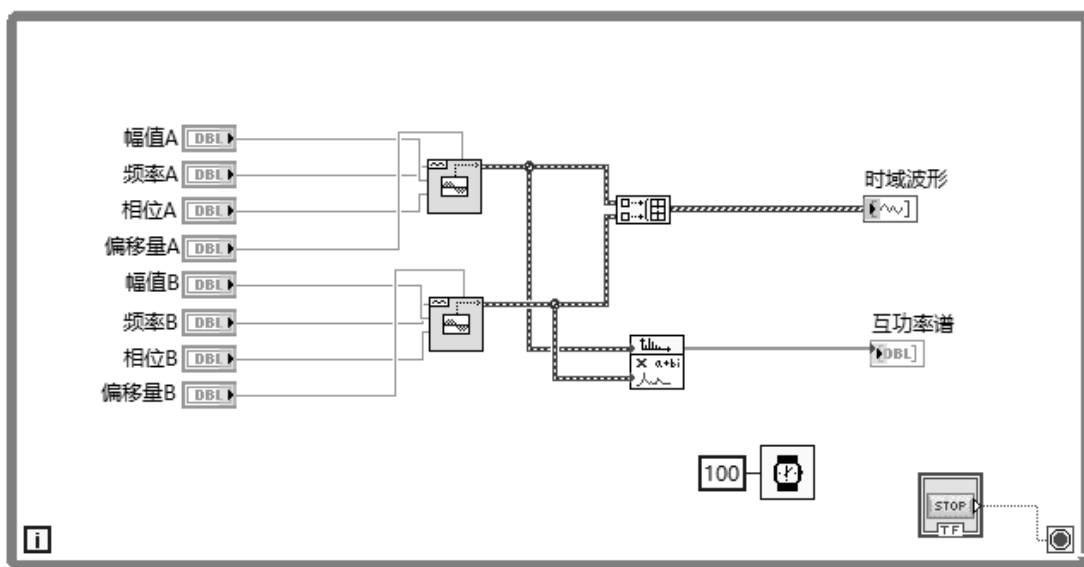


图 12.32 互功率谱程序框图

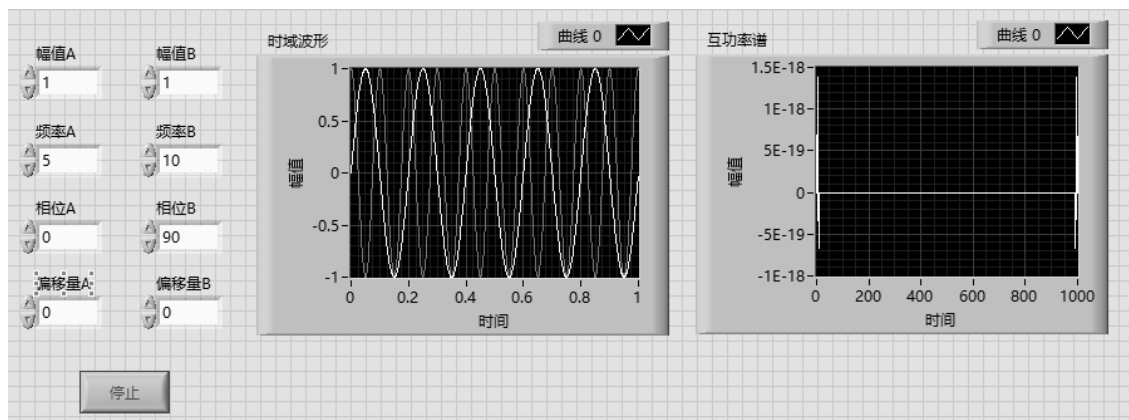


图 12.33 运行结果

## 12.5 信号变换

数字信号处理理论及其技术从诞生发展到现在,随着解决各种信号处理问题的需要,已经产生了许多信号变换方法,各种信号变换的本质实际上都是将原始的时域信号从各种不同的角度进行转换,换一个视角去观察和分析信号中的隐藏信息。

除了最常用的傅里叶变换外,还有许多其他各种类型的信号变换,比如离散余弦变换、离散正弦变换、沃尔什变换、希尔伯特变换、小波变换等,正是这些新变换方法的不断诞生和发展,解决了许多单靠傅里叶变换无法解决的信号处理问题。

12.5.1 信号变换相关的 VI

在 LabVIEW 的信号处理 VI 中，为部分信号变换方法提供了可调用的基本 VI，它们分布在函数选板“信号处理→变换”下，如图 12.34 所示。



图 12.34 信号变换 VI 库

除了拉普拉斯变换 VI 以外，其他 VI 都是以正反变换形式成对出现的，这些 VI 的名称和功能简介如表 12.1 所示。

表 12.1 信号变换 VI 库

VI 名	图 标	功 能
FFT		对输入数组信号进行快速 Fourier 变换
反 FFT		对输入数组信号进行快速 Fourier 反变换
快速希尔伯特变换		对输入数组信号进行快速 Hilbert 变换
反快速希尔伯特变换		对输入数组信号进行快速 Hilbert 反变换
FHT		对输入数组信号进行快速 Hartley 变换
反 FHT		对输入数组信号进行快速 Hartley 反变换
Daubechies4 小波变换		以 Db4 小波为基对输入数组信号进行小波变换
反 Daubechies4 小波变换		以 Db4 小波为基对输入数组信号进行小波反变换
Walsh Hadmard 变换		对输入数组信号进行 Walsh Hadmard 变换
Walsh Hadmard 反变换		对输入数组信号进行 Walsh Hadmard 反变换
离散余弦变换		对输入数组信号进行离散余弦变换
反离散余弦变换		对输入数组信号进行离散余弦反变换
离散正弦变换		对输入数组信号进行离散正弦变换
反离散正弦变换		对输入数组信号进行离散正弦反变换
Chirp Z 变换		对输入数组信号进行 Chrip-Z 变换
反 Chirp Z 变换		对输入数组信号进行 Chrip-Z 反变换
实数拉普拉斯变换		对输入数组信号进行实 Laplace 变换

用户可根据需要对这些 VI 进行调用，当然也可以在此基础上写出自己的信号处理算

法。由于各种信号变换算法的应用十分广泛，篇幅所限，这里只能就其中一两种变换的典型应用举例介绍，旨在抛砖引玉。

## 12.5.2 信号变换举例

例 12-6 Hilbert 变换提取信号包络实例。

信号  $x(t)$  的 Hilbert 变换（希尔伯特变换）的定义为：

$$h(t) = H(x(t)) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau$$

它作为一种数学工具被广泛应用在通信系统或数字信号处理中，如提取瞬时频率和相位信息、提取信号包络以及检测回波等。

LabVIEW 中提供了快速希尔伯特变换（Fast Hilbert Transform）的计算 VI “Fast Hilbert Transform.vi” 与反变换 VI “Inverse Fast Hilbert Transform.vi”。

信号  $x(t)$  与其 Hilbert 变换  $h(t)$  构成一对希尔伯特变换对，信号  $x(t)$  的包络计算式可以表示为：

$$E(t) = \sqrt{x^2(t) + h^2(t)}$$

根据该原理，对一个高斯调制正弦信号进行包络提取，在 LabVIEW 中实现的具体步骤如下。

（1）新建“提取信号包络.vi”，添加“Gaussian Modulated Sine Pattern.vi”，以生成一个高斯调制正弦信号，对参数幅值、中心频率、采样数、时延等创建各自的输入控件，以便进行调节。

（2）添加“Fast Hilbert Transform.vi”计算信号的希尔伯特变换结果，并与原信号组成 Hilbert 变换时，计算出其复数的模值，就是原信号的上包络结果。

（3）由于该信号上下对称，所以可对上包络直接取负后得到信号的下包络，将原信号、上包络和下包络绘制于同一个波形图中。

最终程序的前面板和框图如图 12.35 所示，可以清楚地看到上（下）包络通过原信号波形的每一个波峰（谷）点。当然，对于其他类型的信号，也可以借助此方法求其包络。

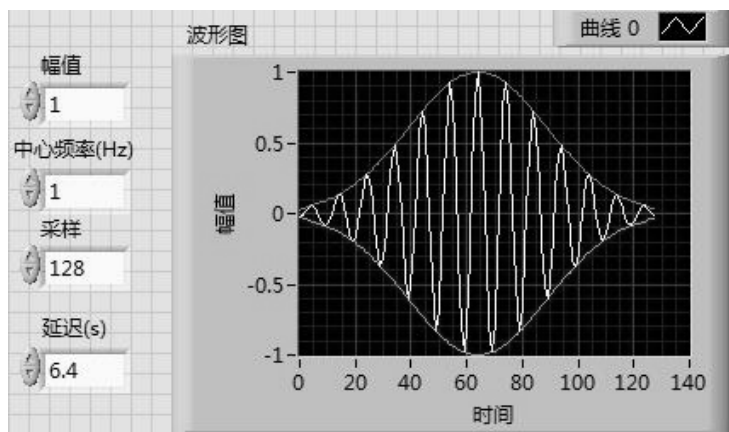


图 12.35 “提取信号包络.vi”的前面板和框图



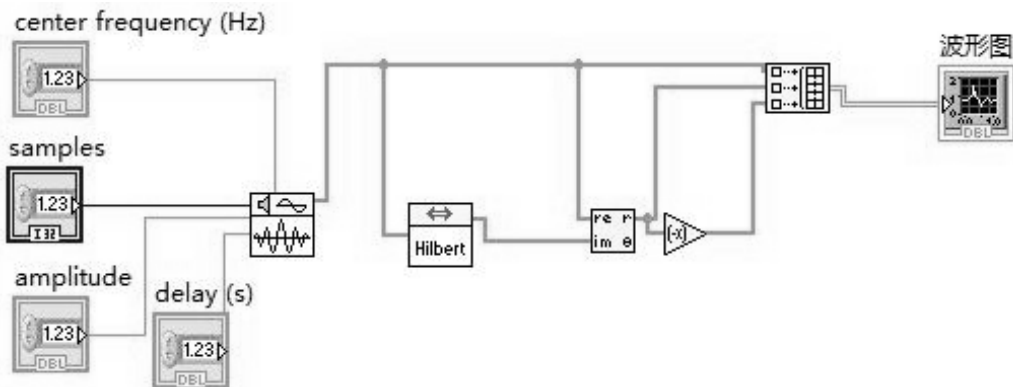


图 12.35 “提取信号包络.vi”的前面板和框图（续）

### 例 12-7 数字滤波器设计。

(1) 启动 LabVIEW 2013，打开程序框图，选择函数选板中“编程→信号处理→波形生成→基本函数发生器”控件，置于程序框图中，在“混合单频”端口右击，创建数组常量，在“噪声 (rms)”端口右击，创建常数 0.5。

(2) 切换到前面板窗口，拖动“混合单频”边框来增加一组频率信号，混合单频信号参数如图 12.36 所示。选择控件选板中“新式→图形→波形图”控件，将两个波形图置于前面板中，分别命名为“原始信号”和“滤波后信号”。选择“新式→布尔→停止按钮”控件，置于前面板。



图 12.36 混合单频信号

(3) 选择“新式→下拉表与枚举→枚举”控件，命名为“滤波器选择”，右击该控件，从弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框中选择“编辑项”选项卡设置枚举值，如图 12.37 所示。

(4) 切换到程序框图窗口，选择“编程→结构→条件结构”控件，将“滤波器选择”控件连接到“条件结构”控件的条件输入端，此时第一个分支“FIR 加窗滤波器”即成为“默认分支”。选择“编程→信号处理→滤波器→高级 FIR 滤波器→FIR 加窗滤波器系数”控件，放置在默认“FIR 加窗滤波器”分支内，x 输入端口与“混合单频”信号输出端口连接，滤波后的 x 与“滤波后信号”的输入端口连接。“混合单频”控件其余端口通过右键“创建→输入控件”来设置，并将其全部置于条件结构体外。“FIR 加窗滤波器”分支结构如图 12.38 所示。

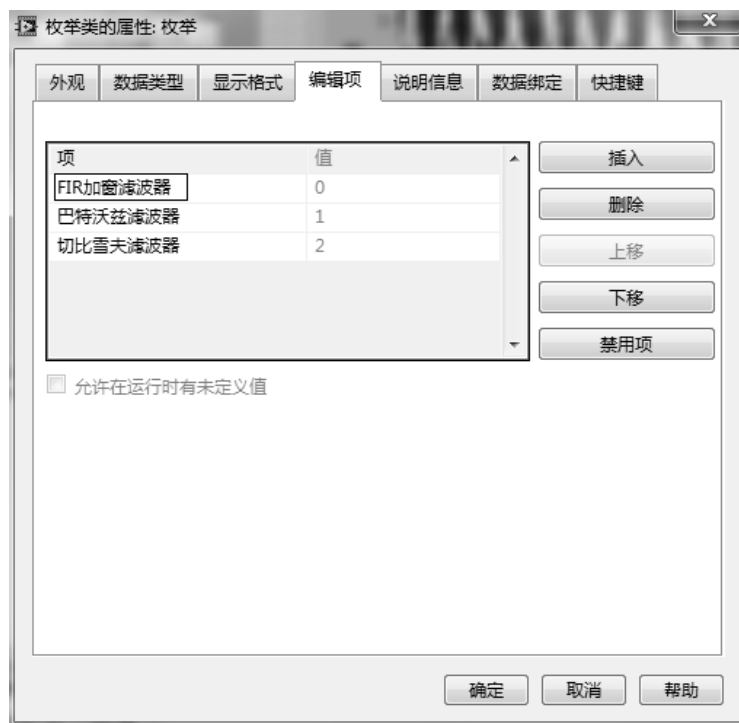


图 12.37 枚举类属性对话框

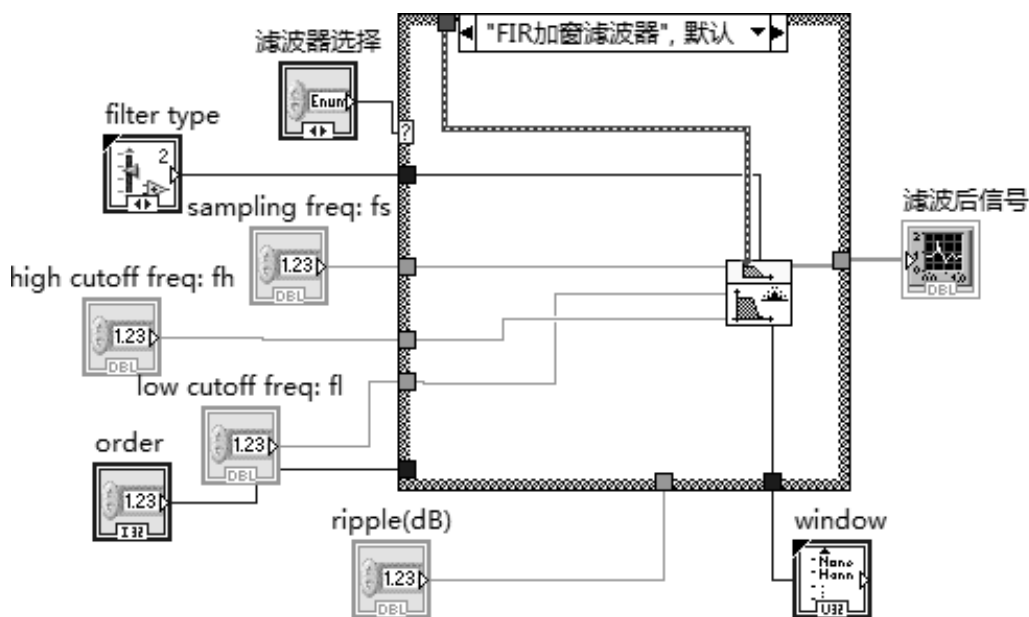


图 12.38 “FIR 加窗滤波器”分支结构

(5) 单击条件结构选择器两侧的黑头，切换到第二分支，用同样的方法添加“巴特沃兹滤波器”，x 输入端口与“混合单频”信号输出端口连接，滤波后的 x 与“滤波后信号”的输入端口连接，其余端口与对应的节点连接。在条件结构体边框上右击。从弹出的快捷菜单

中选择“在后面添加分支”命令。此时“选择条件器”中自动显示“切比雪夫滤波器”，在此分支中添加“切比雪夫滤波器”控件，x 输入端口与“混合单频”信号输出端口连接，滤波后的 x 与“滤波后信号”的输入端口连接，其余端口与对应的节点连接。

(6) 选择“编程→结构→For 循环”控件，将所有控件置于 For 循环体内，将“停止”控件与 For 循环的循环条件相连。

(7) 在切换的前面板中选择滤波器，设置滤波器参数，运行程序。前面板和程序框图如图 12.39 所示。

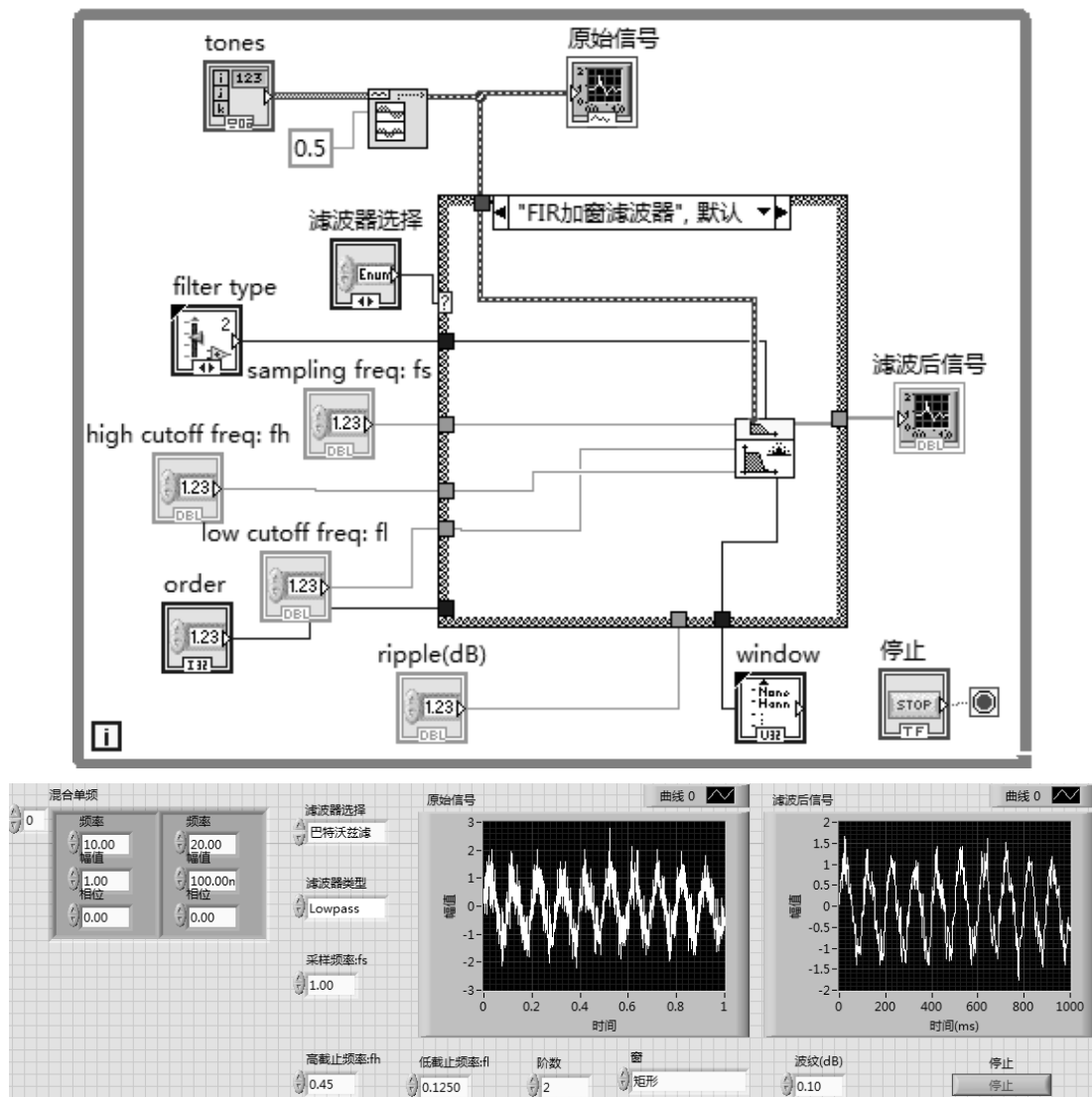


图 12.39 前面板和程序框图

**例 12-8** 对多输入多频信号进行快速傅里叶变换，观察其频率组成成分。

(1) 启动 LabVIEW 2013，打开程序框图，选择函数选板中“编程→信号处理→波形生成→混合单频信号发生器”控件，置于程序框图中，产生一个多频信号；选择“编程→信号

处理→变化→FFT”及“编程→编程→数值→复数至极坐标转换”控件，置于程序框图中，“FFT”的输入端口与多频信号的输出端连接，“FFT”的输出端口与“复数至极坐标转换”的输入端连接。

(2) 选择“编程→编程→数组→数组大小”控件，置于程序框图中，通过动态数据转换与多频信号的输出端连接；选择“编程→编程→数值→除”控件，其两个输入端分别与“复数至极坐标转换”控件的“r”、“数组大小”输出端连接，此时，从“除”输出端输出的信号为双边傅里叶变换信号，程序框图如图 12.40 所示。

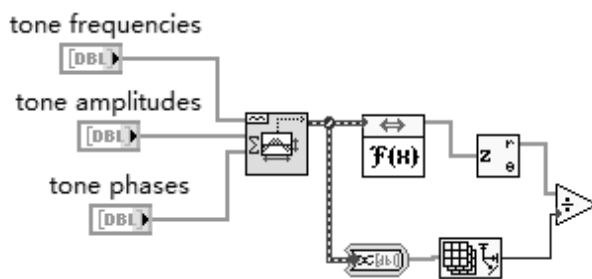


图 12.40 双边傅里叶变换程序图

(3) 选择“编程→编程→数组→索引数组”及“编程→编程→数组→索引子集”控件，选择“编程→编程→数值→商与余数”控件以及其他相应的节点控件，按单边傅里叶变换公式进行连接，最终输出单边傅里叶信号。

(4) 切换到前面板窗口，选择控件选板“新式→图形→波形图”控件，将两个波形图置于前面板中，分别命名为“原始信号”和“变换后信号”。选择“新式→布尔→停止按钮”控件，置于前面板，并在程序框图中连线。

(5) 选择“编程→结构→For 循环”控件，将其置于程序框图中，将所有的函数和节点置于 For 循环体中，“停止”节点与其循环条件连接。

(6) 在前面板中设置输入信号的参数，运行。前面板和程序框图如图 12.41 所示。

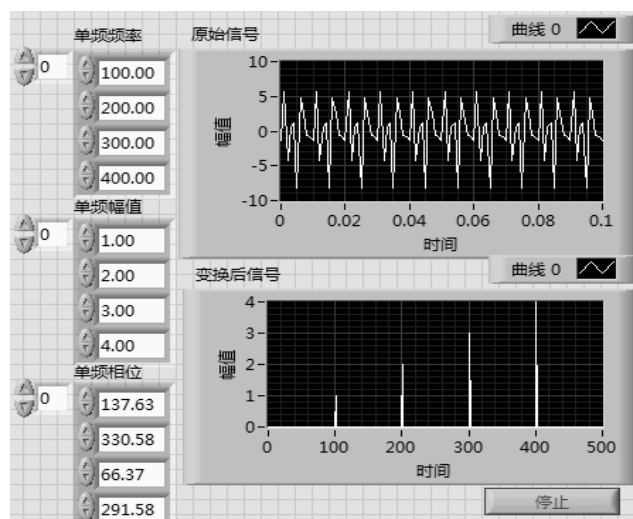


图 12.41 前面板和程序框图

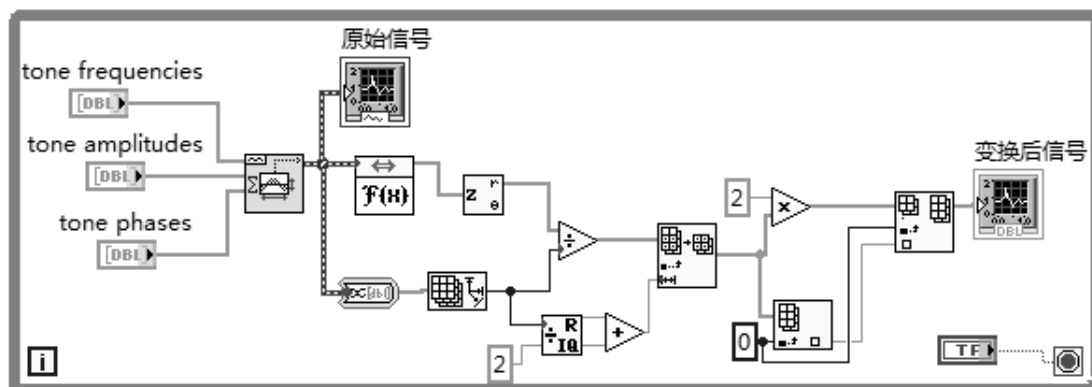


图 12.41 前面板和程序框图 (续)

# 第 13 章 LabVIEW 的应用实例

## 13.1 轧机齿轮箱双通道振动数据采集系统设计

### 13.1.1 设计目的

利用虚拟仪器软件 LabVIEW 完成双通道数据采集系统的设计，实现传统示波器的基本功能，如波形的选择（可以选择正弦波、三角波、方波等各种基本波形），单/双通道的选择，以及对波形各参数的设置（如波形的频率、幅值的更改）。

### 13.1.2 设计要求

- (1) 熟悉传统示波器的基本原理、操作界面各个按钮的作用；
- (2) 能熟练掌握虚拟仪器软件 labVIEW 的基本界面，以及各个元件的属性及作用；
- (3) 能利用 labVIEW 独立完成传统示波器的基本功能，熟悉各程序结构及其实现原理。

### 13.1.3 设计过程

#### 1. 虚拟示波器实现原理

在双通道数据采集回放系统设计中，首先要实现基本波形模拟发生、数据采集记录，其次对记录数据进行回放、分析处理。以下就对每一个部分进行原理介绍。

#### 2. 设计过程及说明

##### 1) 波形发生测量系统

这是一个虚拟的波形发生器，它能实现基本波形的发生，包括方波、正弦波、锯齿波等，用循环控制波形发生的频率。

要建立一个实用的数据采集系统，必须了解一些关于模拟信号采集过程和模拟输入系统设计的基本知识。根据信号的特征和测试目的，模拟信号可以分为 3 类：

- (1) 对于随时间缓慢变化的信号，例如容器的液位、对象的温度等，通常叫作直流信号。对直流信号一般只需要比较慢的采样频率。

(2) 对于随时间变化较快的信号, 如果需要了解它的波形, 则把它作为一个时域信号来处理, 这时候就需要比较高的采样频率。

(3) 对于随时间变化较快的信号, 如果需要了解它的频率成分, 则把它作为一个频域信号处理。要得到准确的频率信息, 采样率必须大于信号最大频率成分的两倍。采样率的一半叫奈奎斯特频率。这实际上意味着对于最大频率的信号成分, 每一个周期只采样两个数据点, 这对于描述信号波形是远远不够的。在工程实际中, 一般使用信号最高频率成分 4~10 倍的采样率。

波形发生 VI 的程序结构采用 case 结构。channel 值对应 0: 无波形发生; 1: 方波发生; 2: 正弦波发生; 3: 三角波发生。分别如下图 13.1、图 13.2、图 13.3 所示。Vol 为幅值, Fre 为频率。

通道 2 与通道 1 类似。

双通道数据采集系统 VI 的调用连接如图 13.4 所示。

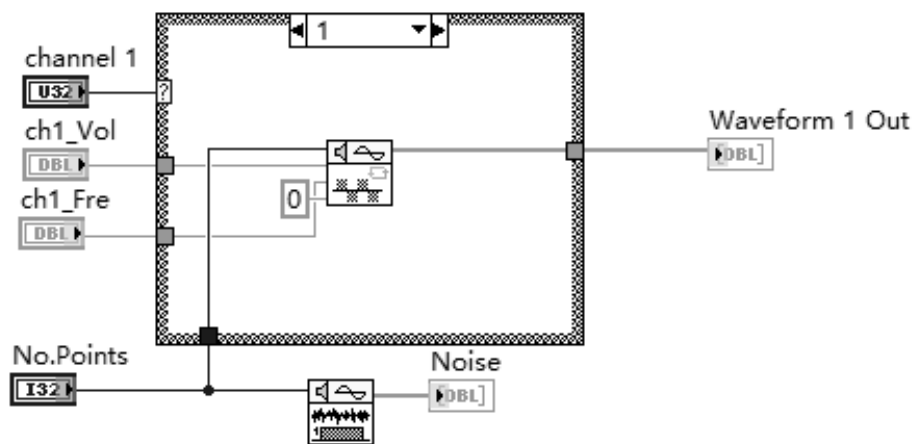


图 13.1 方波发生程序框图

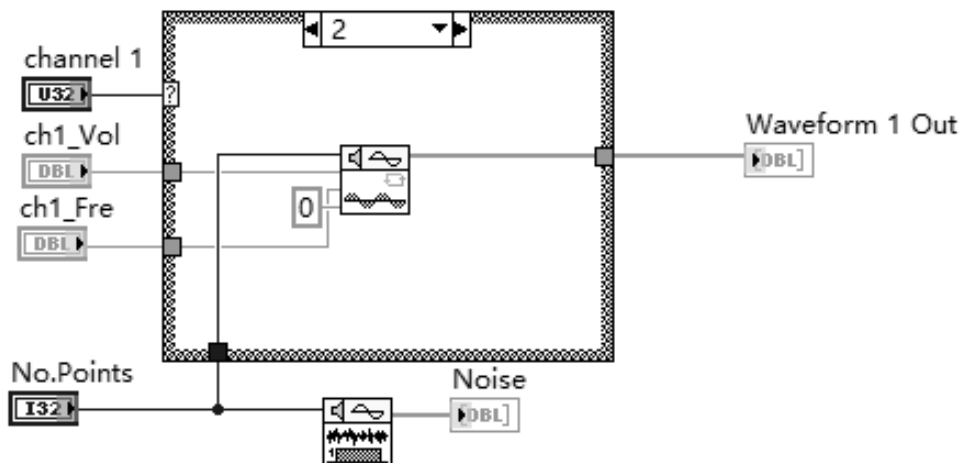


图 13.2 正弦波发生程序框图

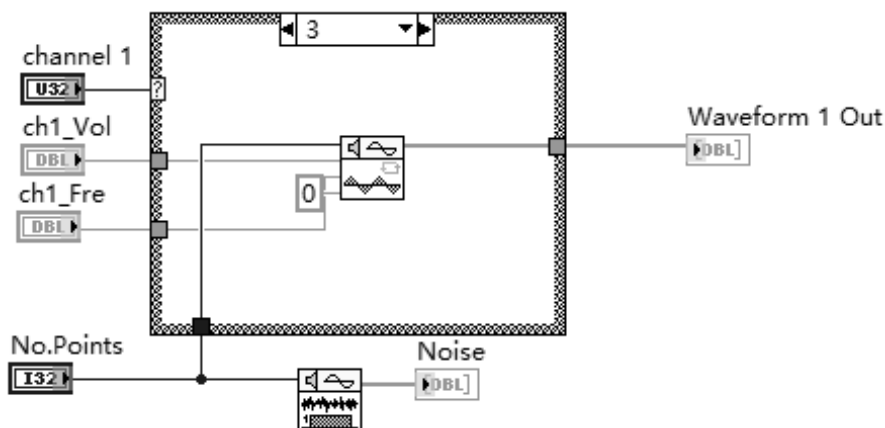


图 13.3 三角波发生程序框图

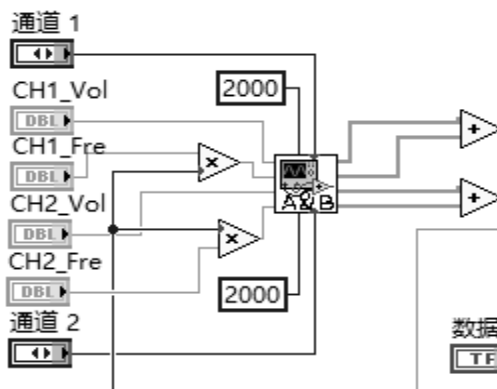


图 13.4 双通道数据采集系统 VI 的调用连接

时间基准、电压基准的设置：都采用的是 case 结构。

时间基准对应前面板的 5ms/Div、10ms/Div、15ms/Div 的 case 分支，程序框图分别如图 13.5 至图 13.7 所示。

电压基准对应前面板的 0.5V/Div、1V/Div、2V/Div 的 case 分支，程序框图分别如图 13.8 至图 13.10 所示。

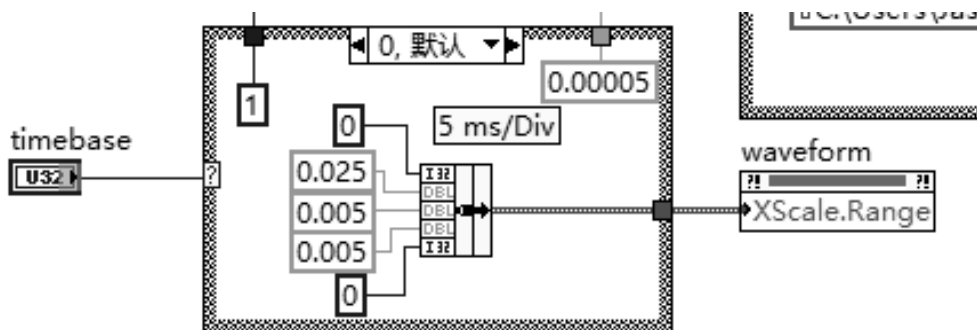


图 13.5 5ms/Div 的 case 分支



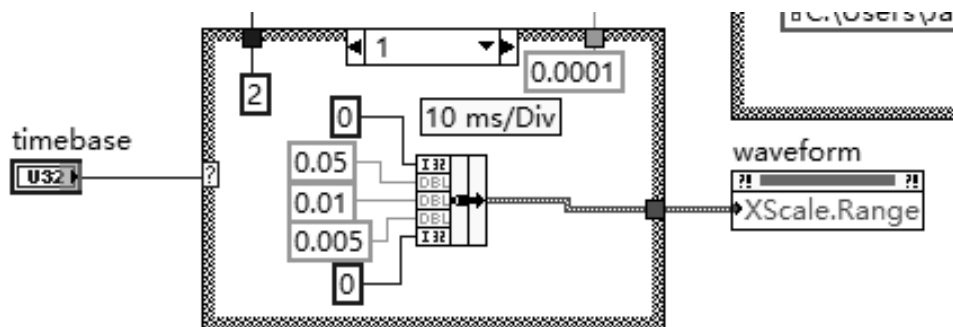


图 13.6 10ms/Div 的 case 分支

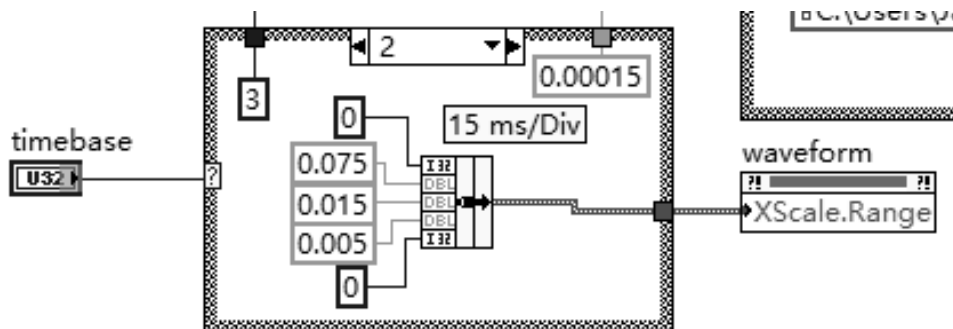


图 13.7 15ms/Div 的 case 分支

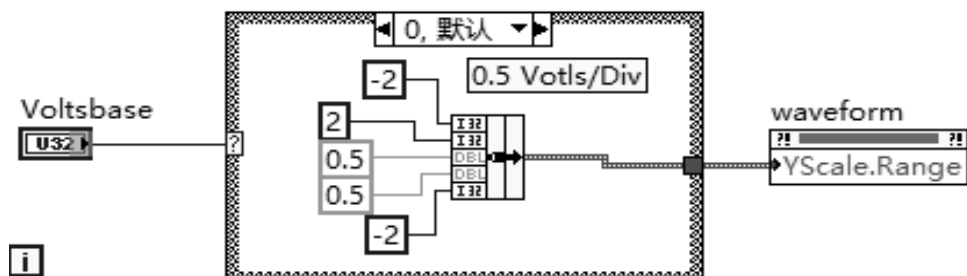


图 13.8 0.5V/Div case 分支

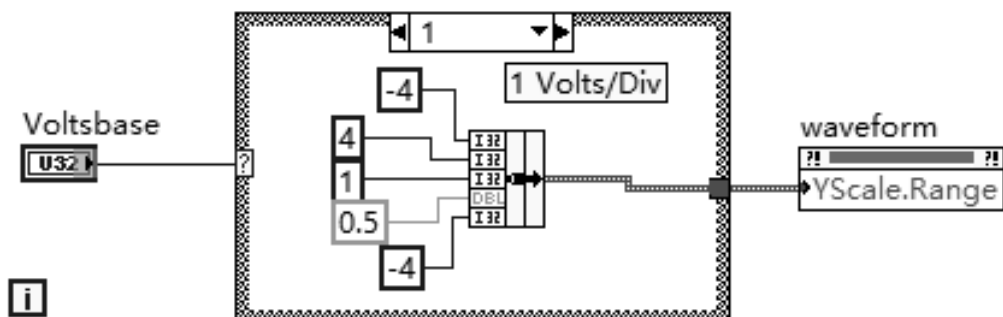


图 13.9 1V/Div 的 case 分支

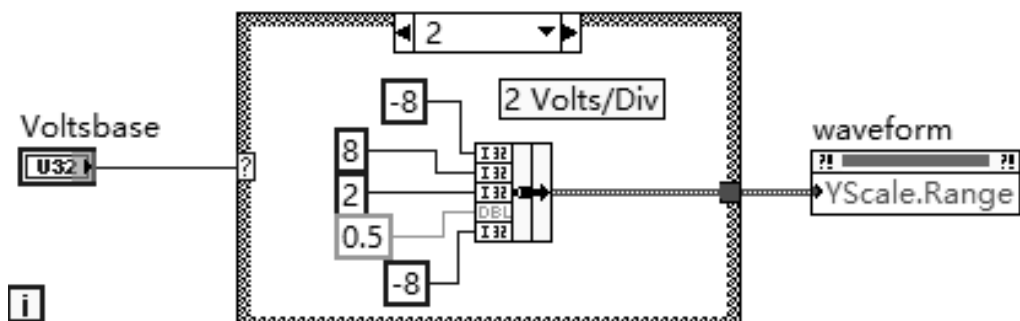


图 13.10 2V/Div 的 case 分支

## 2) 频谱分析

信号处理几乎涉及所有的工程技术领域，而频谱分析正是信号处理中一个非常重要的分析手段。一般的频谱分析都依靠传统频谱分析仪来完成，这些设备价格昂贵，体积庞大，不便于工程技术人员的携带。虚拟频谱分析改变了原有频谱分析的整体设计思路，用软件代替了硬件，使工程技术人员带一部笔记本电脑到现场就可轻松完成信号的采集、处理及频谱分析。

**时间域分析：**它是最直观的分析。从时域分析中既可做出一些原始判断，又可确定进一步分析的方向和目标。

**频谱分析：**将时域信号变换成频域信号再分析称为频谱分析。由于时域信号分为连续信号和离散信号，连续信号又可分为绝对可积、平方可积和均方可积；离散信号又可分为绝对可和、平方可和和均方可和，故对应的频谱可分为多种。

**时域加窗：**时域加窗可减少泄露，还有其他用途，加窗可加矩形窗、海窗、平顶窗、力窗和指数窗等，注意每种窗都是既有优点又有缺点。

**矩形（None）窗（也叫均匀窗、不加窗）：**它的功能仅为以采样时间为窗长度截断原始信号，它的泄露较大，仅用于无泄露场合。

**海窗包括汉宁窗（Hanning）和汉明（Hamming）窗：**用于减少泄露，缺点在于频域主瓣比矩形窗主瓣宽，确定峰值频率时误差较大。加海窗会降低峰值高度。

**平顶窗（Flat top）**主要用于提高分析仪的幅值读数精度。输入一个峰值已知的正弦波，用平顶窗在频域读数，可发现它的读数最接近于真正峰值。

**布拉克曼窗（Blackman）：**为了更进一步抑制旁瓣，可再加上余弦的二次谐波分量，得到布拉克曼窗。布拉克曼窗的旁瓣衰减加大，但同时主瓣宽度也相应加宽了。

在本程序设计中，此模块处理分析为回放 VI，其程序框图如图 13.11 所示。

首先从文件里读取数据，读取的原始数据用示波器 1 显示出来；然后通过滤波器，滤波器用了 case 结构，可以选择不同的滤波器，滤波后的数据通过示波器 2 显示；滤波后的数据又通过自相关时域分析处理，再经过加窗函数，其中加窗函数也用了 case 结构，可以选择不

同的加窗类型，最后加窗后的数据再经过频域分析傅里叶变换，得到最终的频谱分析数据经过示波器 3 显示出来。

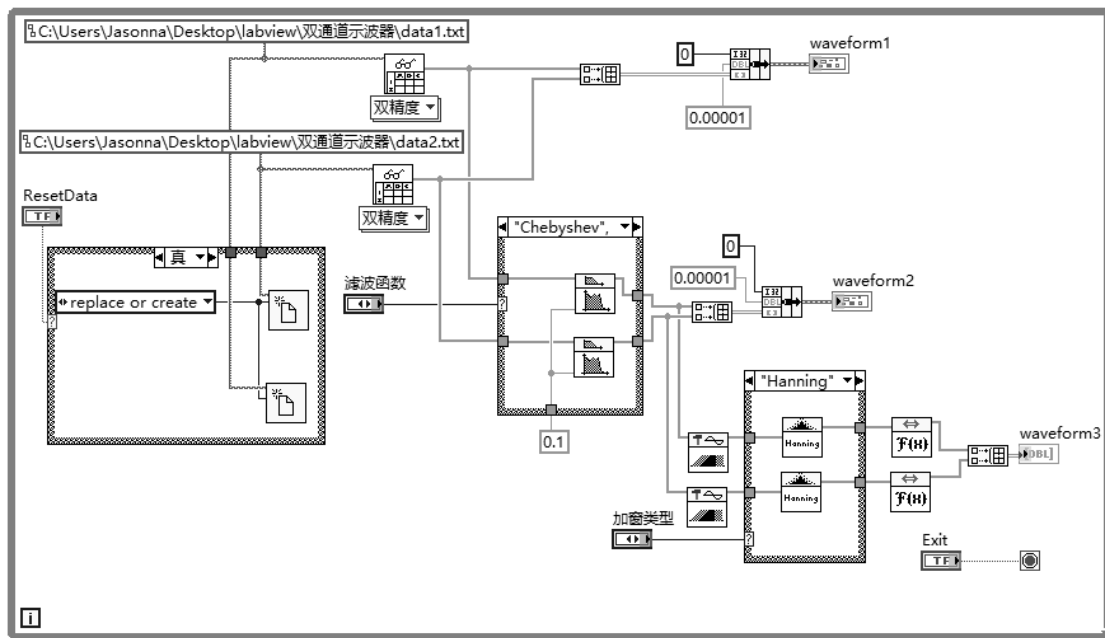


图 13.11 回放 VI 程序框图

### 3) While 结构实现虚拟示波器

While 循环可以反复执行循环体的程序，直至到达某个边界条件。它类似于普通编程语言中的 Do 循环和 Repeat-Until 循环。While 循环的框图是一个大小可变的方框，用于执行框中的程序，直到条件端子接收到的布尔值为 FALSE。

该循环有如下特点：

- (1) 计数从 0 开始 ( $i=0$ )。
- (2) 先执行循环体，而后  $i+1$ ，如果循环只执行一次，那么循环输出值  $i=0$ 。

循环至少要运行一次。

(3) While 循环的执行次数是不确定的，只要设置的条件为真，循环程序就会持续运行。为了不占用大量的内存，在 While 循环中添加适量的等待延时。

### 4) 虚拟示波器完整程序图

轧机齿轮箱双通道数据采集处理系统最终的系统框图如图 13.12 所示，图 13.13 为轧机齿轮箱双通道数据采集处理系统前面板。

图 13.14、图 13.15 为运行后结果显示与数据回放前面板。

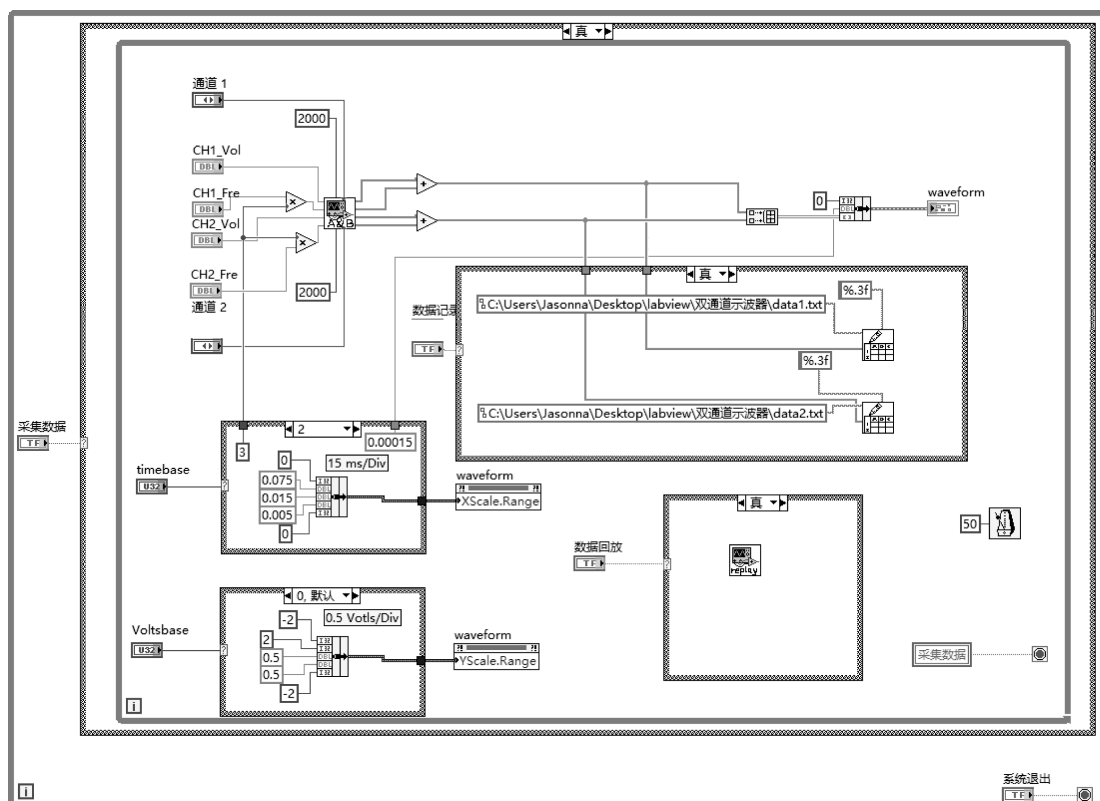


图 13.12 轧机齿轮箱双通道数据采集处理系统框图

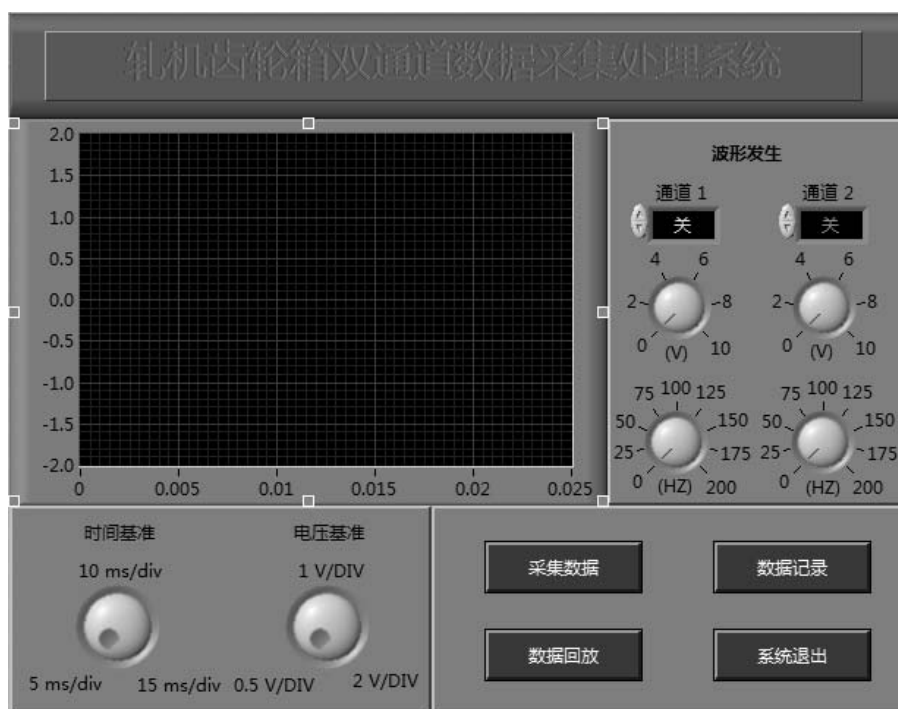


图 13.13 轧机齿轮箱双通道数据采集处理系统前面板



图 13.14 运行结果显示

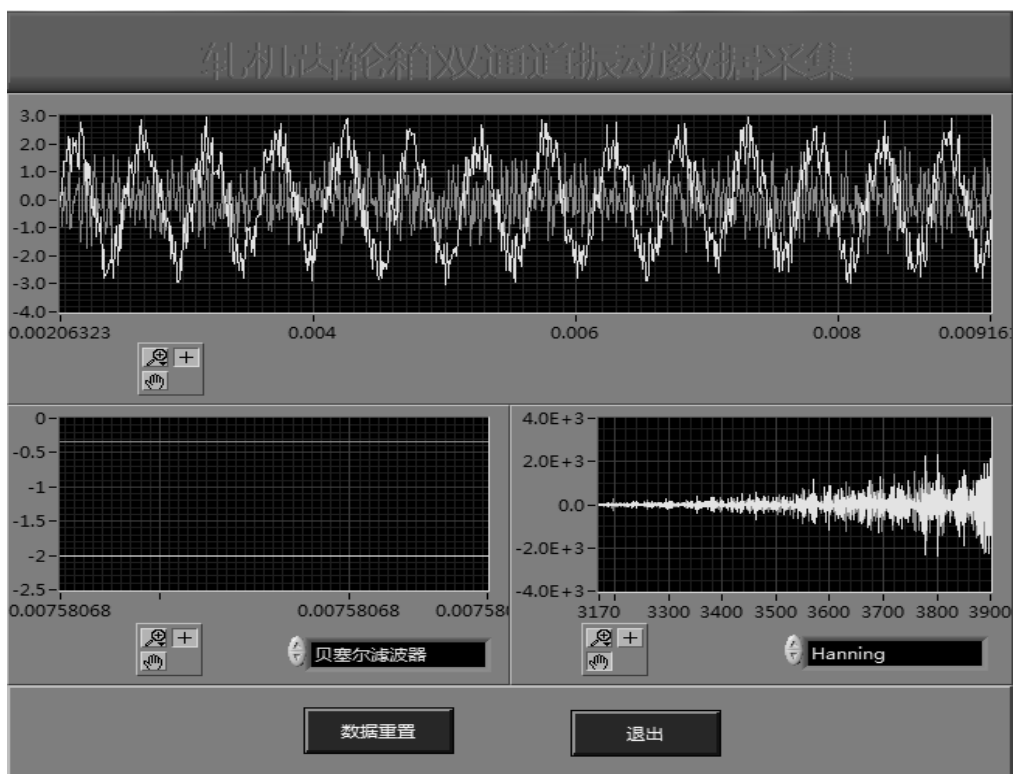


图 13.15 数据回放的前面板

## 13.2 风机齿轮箱监测与分析系统设计

本节以风机齿轮箱振动测试分析为例，详细介绍使用 LabVIEW 软件设计振动信号采集与分析系统的全过程，包括前面板中数据采集的参数设置、程序结构的构架等内容。

### 13.2.1 设计目的

齿轮箱作为机械设备中一种必不可少的连接和传递动力的通用零部件，有着广泛的应用，发挥着重要的作用。齿轮箱本身结构复杂，工作环境恶劣，极易受到损伤和出现故障。故障隐患将影响到生产运行的经济性和安全性。为了保证齿轮箱能够安全、可靠、稳定地工作，需要对其状态进行监测与分析。整个系统要考虑可靠性、经济性、实用性等要求。因此，本例以 LabVIEW 虚拟平台为基础，使用图形语言编辑，设计了基于 LabVIEW 的风机齿轮箱监测与分析系统。

本例能够实现实验数据的采集、显示、分析等功能。具有界面直观、功能齐全、交互界面友好等特点。

### 13.2.2 系统的硬件构成

数据采集与分析系统由传感器、信号调理电路、数据采集卡、计算机等部分组成。振动信号是机械设备状态信息的载体，是进行故障诊断的基础，在故障诊断中最重要的信息就是设备的振动信号。另外，由于风机运行于不稳定的工况条件下会导致风机齿轮箱的转速变化，故要准确全面地进行信息提取，需要采集齿轮箱的振动信号。在数据采集的硬件实现过程中主要包括两部分内容：传感器选型和安装，数据采集设备的选型和配置。

#### 1) 传感器的选择

在振动测量中，使用最多的是压电式加速度传感器，压电式加速度传感器的输出是与被测物体振动加速度成正比的电量，它的主要优点是体积小、质量轻、频率范围宽、比较稳定、安装方便。本系统中选用 PCB352C33 压电式加速度计。加速度传感器是接触式传感器，使用时要把传感器安装在被测对象上，传感器与被测对象一起振动，加速度传感器可采用螺栓安装的方式或粘贴的方式。加速度传感器在安装时要注意以下几点：保证传感器与被测对象之间接触面比较平滑，选择合理的方法安装传感器，检测点应尽量靠近故障源。

#### 2) 采集卡的选择

数据采集卡的功能是把模拟信号转换为数字信号，并把采集到的信号传输给计算机。计算机通过它来采集和处理模拟信号，数据采集卡是数据采集硬件的核心。

本系统采用的 NI9234，它是 4 通道 C 系列同步动态信号采集模块，支持 IEPE 与非 IEPE 传感器的连接，可以进行高精度振动/音频信号测量。NI9234 具有 102dB 动态范围，同时对

IEPE 加速度传感器提供信号调理和交流耦合，其采集通道提供可以自动调节采样率的内置抗混叠滤波器，同时每通道可高速实现对信号的数字化。

### 13.2.3 系统软件设计

#### 1. 系统前面板的设计

本系统的前面板主要由参数设置、显示、分析等部分组成。参数设置主要包括通道配置参数和采集设置。通道配置包括各个传感器所对应的通道，以及采集数据的最大值、最小值等。采集参数需要设置采样模式、采样率和每通道采样的数据个数以及传感器的灵敏度、单位等相关参数。采样模式包括有限采样、连续采样和硬件定时单点采样三种选择。显示主要对采集的数据以图形的方式进行实时显示。分析部分是对信号进行功率谱分析。

根据界面的整体设计规划，需要在前面板上布置多个面板，用于实现不同调节功能和控制功能。需要布置的前面板主要包括：通道选择面板、采集设置面板和传感器参数面板。

系统的前面板如图 13.16 所示。

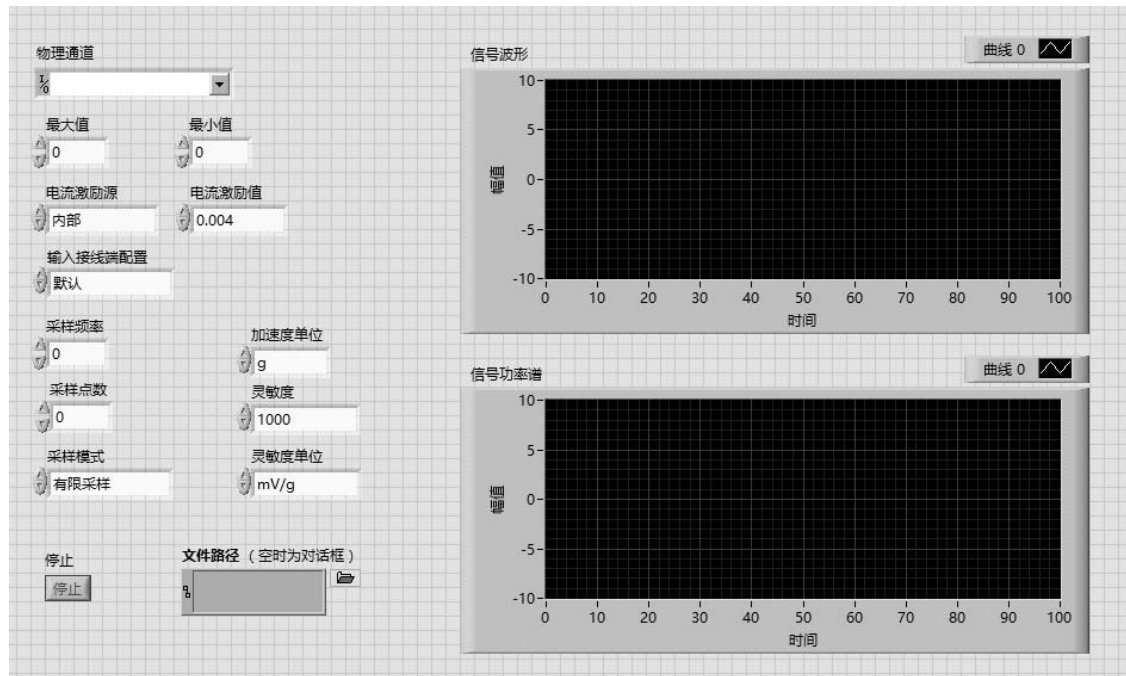


图 13.16 系统前面板

具体实现步骤如下：

- (1) 在启动主界面的“新建”项中创建新 VI，命名为振动分析。
- (2) 进行通道配置。

执行“控件→新式或银色→I/O→DAQmx 名称控件→DAQmx 物理通道（银色）”操作，添加“物理通道”控件；执行“控件→新式→数值→数值输入控件”操作，添加 6 个数值输入控件，分别命名为物理通道最大值、最小值、电流激励源、电流激励值、输入接线端配置，如图 13.17 所示。



图 13.17 通道配置

### (3) 采集参数设置。

执行“控件→新式→数值→数值输入控件”操作，添加三个数值输入控件，分别命名为：采样频率，采样点数，采样模式（有限采样），如图 13.18 所示。

### (4) 传感器参数设置。

执行“控件→新式→数值→数值输入控件”操作，添加三个数值输入控件，分别命名为加速度单位，灵敏度，灵敏度单位，如图 13.19 所示。

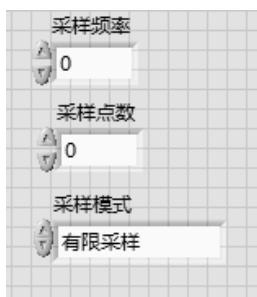


图 13.18 采集参数设置



图 13.19 传感器参数设置

### (5) 信号显示。

执行“控件→新式→图形→波形图”操作，添加两个波形图控件分别命名为“时域波形”、“信号功率谱”，如图 13.20 所示。

### (6) 数据存储。

执行“控件→新式→字符串与路径→文件路径输入”操作，添加“文件路径输入”控件。

(7) 执行“控件→新式→布尔→停止按钮”操作，添加一个“停止”按钮，用于停止运行中的 VI 程序。

## 2. 系统程序框图设计

数据采集和处理分析是本设计的核心内容。数据采集通过 NI-DAQmx 函数来实现。采用“写入电子表格文件”可将采集到的数据保存到文件中。

### (1) 创建单个或多个虚拟通道，并将其添加至任务。

执行“函数→测量 I/O→DAQmx-数据采集→DAQmx 创建虚拟通道”操作，放置到后面板上。鼠标单击该函数右下角的选择器→下拉列表选择“模拟输入→加速度→加速度计”结



果。如图 13.21 和图 13.22 所示。

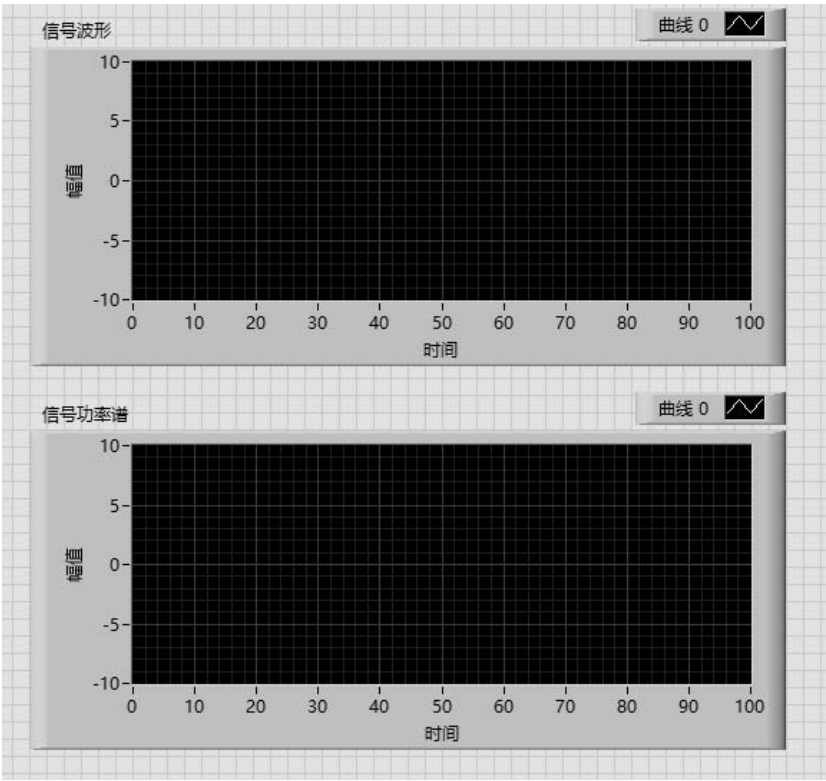


图 13.20 信号波形显示与分析

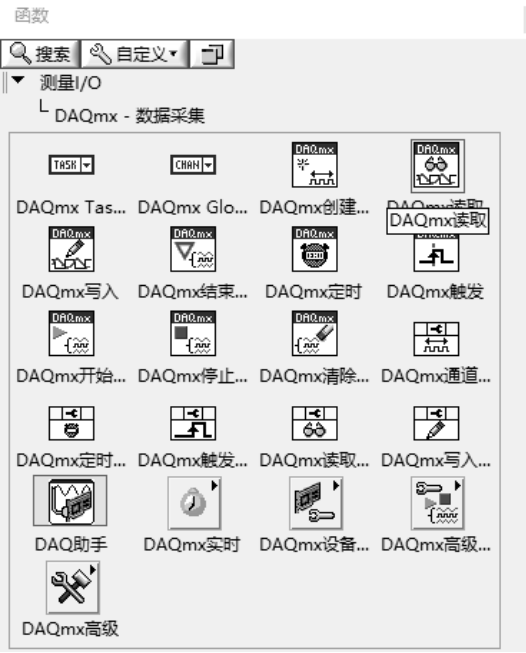


图 13.21 DAQmx 数据采集子选板



图 13.22 DAQmx 创建虚拟通道函数

“输入接线端配置”端子设置为长整型，连接“输入接线端配置”端口；“最小值”端子设置为双精度型，连接“最小值”端口；“最大值”端子设置为双精度型，连接“最大值”端口；“物理通道”端子连接“物理通道”端口；“加速度单位”端子设置为长整型，连接“单位”端口；“灵敏度”端子设置为双精度，连接“灵敏度”端口；“灵敏度单位”端子设置为长整型，连接“灵敏度单位”端口；“电流激励源”端子设置为长整型，连接“电流激励源”端口；“电流激励值”端子设置为双精度型，连接“电流激励值”端口，如图 13.23 所示。

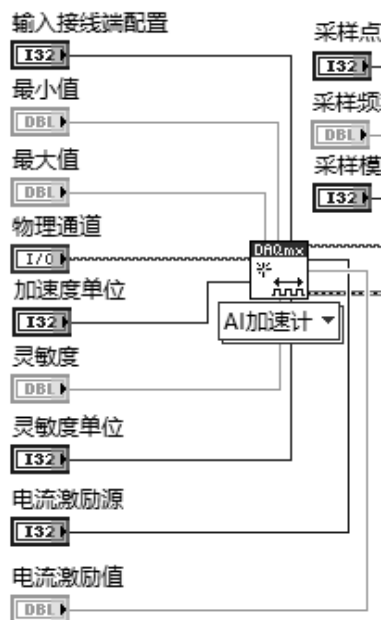
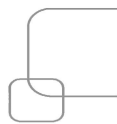


图 13.23 DAQmx 创建虚拟通道设置



(2) 配置要获取或生成的采样数。

执行“函数→编程→测量 I/O→DAQmx-数据采集→DAQmx 定时”操作，添加“DAQ 定时”函数。单击输入控件端子，然后单击鼠标右键出现下拉列表，选择表示法，然后按要求选择数据类型。“采样点数”端子与“每通道采样”端口连接；“采样频率”端子连接“速率”端口；“采样模式”端子连接“采样模式”端口。DAQmx 创建通道连接：“任务输出”端口与 DAQmx 定时函数的“任务/通道输入”端口连接；“错误输出”端子连接对应的“错误输入”端口。

(3) 创建所需缓冲区的程序图。

执行“函数→编程→测量 I/O→DAQmx-数据采集→DAQmx 高级任务选项→DAQmx 配置输入缓冲区”操作，添加“DAQ 配置输入缓冲区”函数。“采样点数”端子连接到“DAQ 配置输入缓冲区”函数的“缓冲区大小（采样通道）”端口；DAQmx 定时器的“任务输出”端口连接 DAQmx 配置输入缓冲区“任务/通道输入”端口；“错误输出”连接对应的“错误输入”端口。

(4) 使任务处于运行状态，开始测量或生成的程序框图。

a. 执行“函数→编程→测量 I/O→DAQmx-数据采集→DAQmx 开始任务”操作，添加“DAQmx 开始任务”函数。DAQmx 配置输入缓冲区的“任务输出”端口连接 DAQmx 开始任务的“任务通道输入”端口；“错误输出”连接相应的“错误输入”端口。

b. 执行“函数→编程→I/O 测试→DAQmx-数据采集→DAQmx 读取”操作，添加“DAQmx 读取”。鼠标单击按右键出现下拉菜单，选择“选择类型”，然后选择“模拟→多通道→多采样→1D 波形”。“DAQmx 读取”控件上的“每通道采样”连接 While 循环外的“采样点数”；“DAQmx 开始任务”的“任务输出”端口与“DAQmx 读取”的“任务/通道输入”端口连接，“错误输出”端口连接相应的“错误输入”端口。

(5) 波形显示、转换与存储。

a. 将“DAQmx 读取”函数上的“数据”端口的输出与信号波形节点端口相连，实现时域信号的显示。

b. 对波形进行转换并输出程序。执行“函数→编程→信号处理→谱分析→功率谱”操作，添加“功率谱”函数，将“DAQmx 读取”控件上的“数据”端口与“功率谱”函数的“X”端口连接，“功率谱”函数的输出与信号功率谱波形端子相连接，对信号进行功率谱分析。

c. 执行“函数→编程→文件 I/O→写入电子表格文件”操作，添加“写入电子表格文件”函数，将 DAQmx 读取”控件上的“数据”端口分别与“写入电子表格”控件的“一维数据”端口连接，“写入电子表格文件”端口文件路径连接“文件路径”控件，实现数据的存储。

(6) 清除任务。

a. 执行“函数→编程→测试 I/O→DAQ-mx 数据采集→DAQmx 清除任务”操作，添加“DAQmx 清除任务”函数。“DAQmx 读取”的“任务输出”端口与“DAQmx 清除任务”的

“任务输入”端口相连，“错误输出”端口与“错误输入”端口连接。

b. 执行“函数→编程→对话框与用户处理→简易错误处理器”操作，添加“简易错误处理器”函数。将“DAQmx 清除任务”的“错误输出”端口与“简易错误处理器”的“错误输入”相连。

(7) 添加循环控制语句。执行“函数→编程→结构→While 循环”操作，添加“While”循环。将 DAQmx 读取、波形显示、转换与存储内容包含在内。

(8) 执行“函数→编程→簇、类与变体→按名称解除捆绑”操作，添加“按名称解除捆绑”函数。鼠标单击“按名称解除捆绑”控件左键，选择 status。

(9) 执行“函数→编程→布尔→或”操作，添加“或”操作函数。

(10) 将“DAQmx 读取”控件上的“错误输出”连接“按名称解除捆绑”函数，“按名称解除捆绑”和“停止”端子与“或”端子进行“或”运算，“或”运算结果连接 While 循环的终止条件，实现循环的控制。

最终的系统主程序框图如图 13.24 所示。

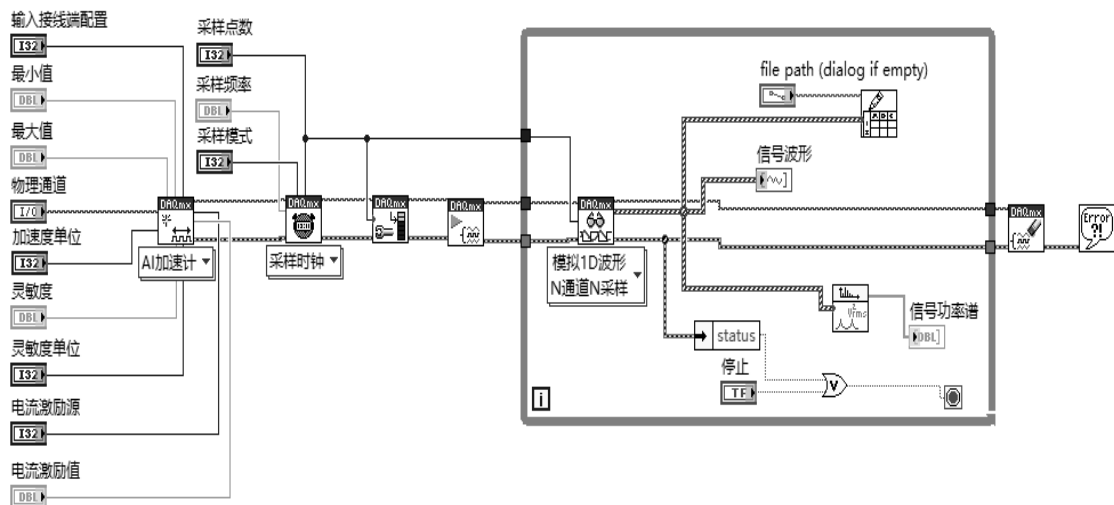


图 13.24 系统主程序框图

## 13.3 矿井通风机轴承振动和温度监测系统

### 13.3.1 系统的总体设计方案

为监测矿井通风机健康状态，设计了矿井通风机的轴承振动、温度在线监测系统。其工作原理：振动信号和温度信号经传感器采集进入信号调理器，转换为标准电信号后进入 PCI 采集卡；上位机中矿井通风机轴承振动和温度在线监测系统软件主要是采集矿井通风机的轴

承振动信号、温度信号，并呈现振动曲线、温度变化显示，对超出预警值的幅值与温度进行报警。

LabVIEW 是一种图形化编程语言，与常见的编程语言相比，其他语言都是文本语言，不直观，对系统的使用者专业水平要求较高，一种图形化的编程系统对实际使用是很有帮助的，因此本系统就是基于图形化编程，用 LabVIEW 中的事件结构把系统的各个功能编写在对应的事件里，通过改变相应操作按钮的值来响应相对应的事件；也可以把系统的各个功能部分分别编写成子程序，然后在主程序中调用各个子程序来完成矿机通风机各项性能参数的监测功能。本系统的软件程序正是把这两种编程方式结合起来编写的，充分发挥了检测系统优越的功能。它不仅能满足数据采集要求，而且还具有操作简单、界面直观、可靠性高、维护简便等优点。

### 13.3.2 前面板设计

系统前面板具体操作步骤如下所述。

(1) 在电脑“开始”菜单查询 NI LabVIEW，打开 LabVIEW 软件，如图 13.25 所示。

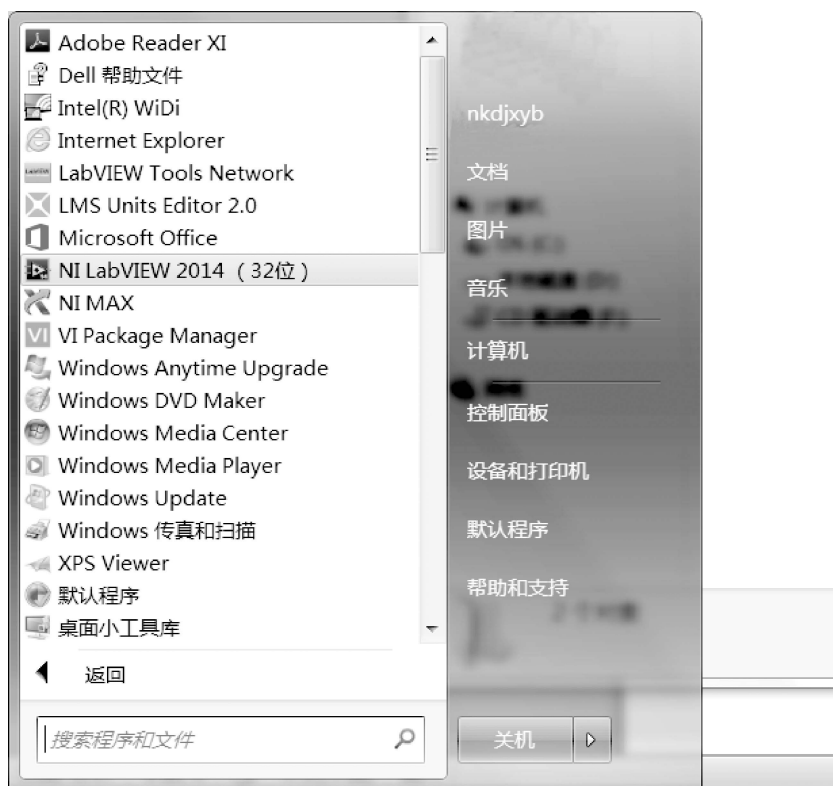


图 13.25 打开 LabVIEW 软件

(2) 单击创建项目，新建 VI，如图 13.26 所示。

新建 VI 的前面板与程序框图，在前面板中打开控件选板，如图 13.27 所示。分别在新

控件中的数值、布尔与图形等子选板选取所需控件。

(3) 从选板中选取所需的选项，拖到前面板上，具体显示如图 13.28 所示。

(4) 打开工具选板，选择自动选择工具。单击图标显示为绿色即可，如图 13.29 所示。

(5) 对拖曳后的控件进行整理，最终结果如图 13.30 所示。



图 13.26 新建 VI



图 13.27 控件选板

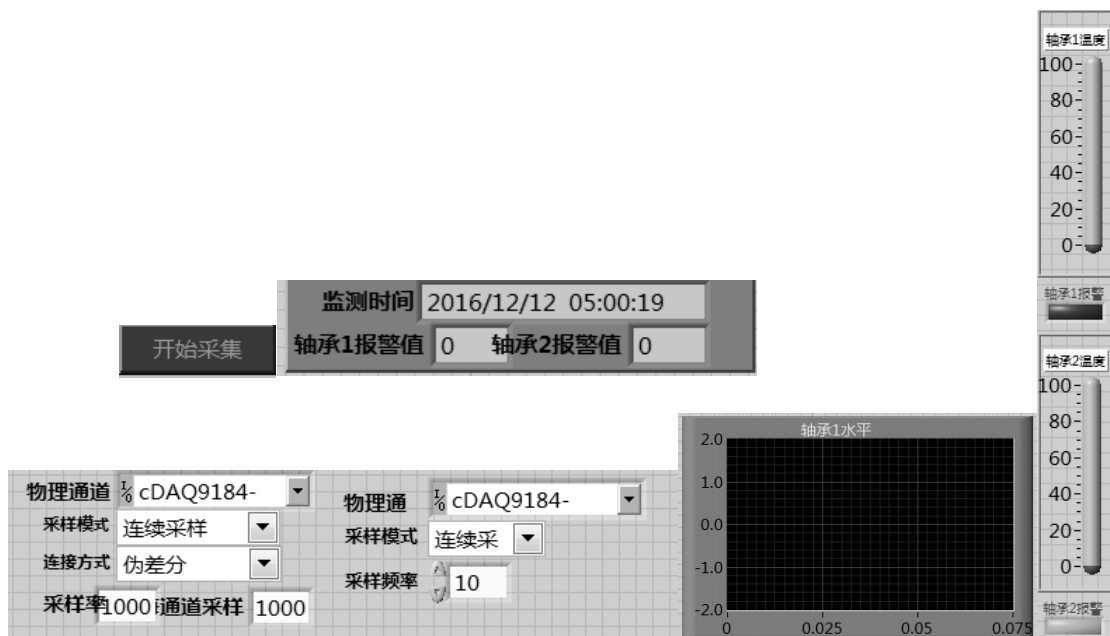


图 13.28 所需控件



图 13.29 激活的工具板

### 13.3.3 程序框图设计

(1) 按快捷键“Ctrl+E”打开程序框图,如图 13.31 所示。

(2) 打开“函数选板”选取一定的数值、结构、测量 I/O 与信号处理等控件, 控件的具体位置如图 13.32 所示。

(3) 选取的温度、振动结构控件等如图 13.33 所示。

(4) 打开工具选板, 对控件“监测时间”进行连线, 结果如图 13.34 所示。

(5) 振动信号采集模块的设置与连线, 选取测量 I/O 的 DAQmx 控件及各种数值控件, 如图 13.35 所示。

(6) 振动信号采集所设计的子 VI 及 DAQmx 通道采集汇总 VI 见图 13.36。

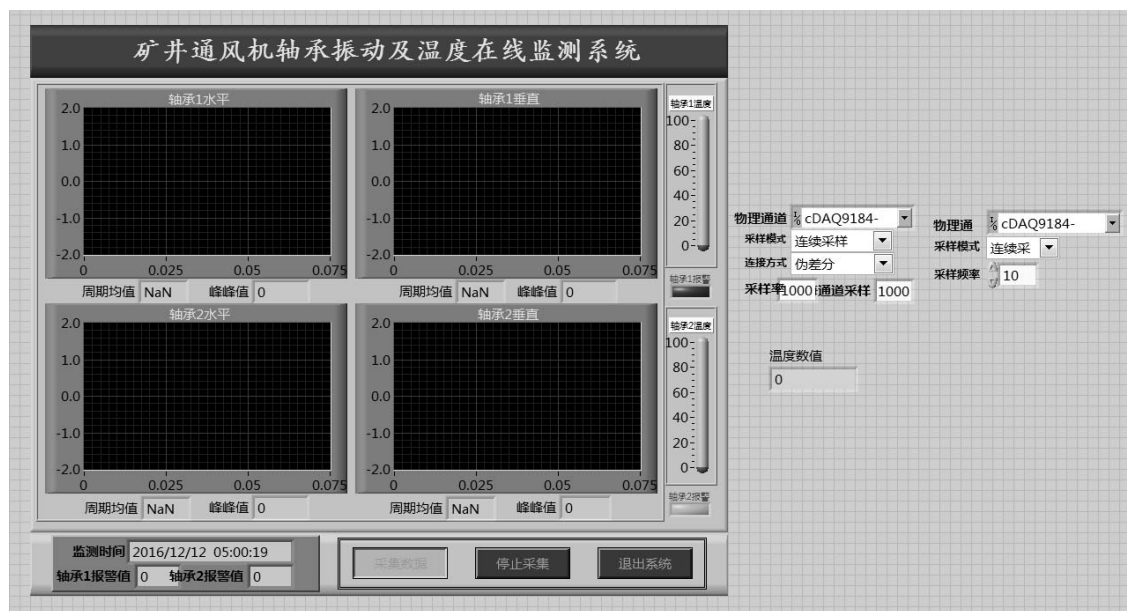


图 13.30 整理后控件

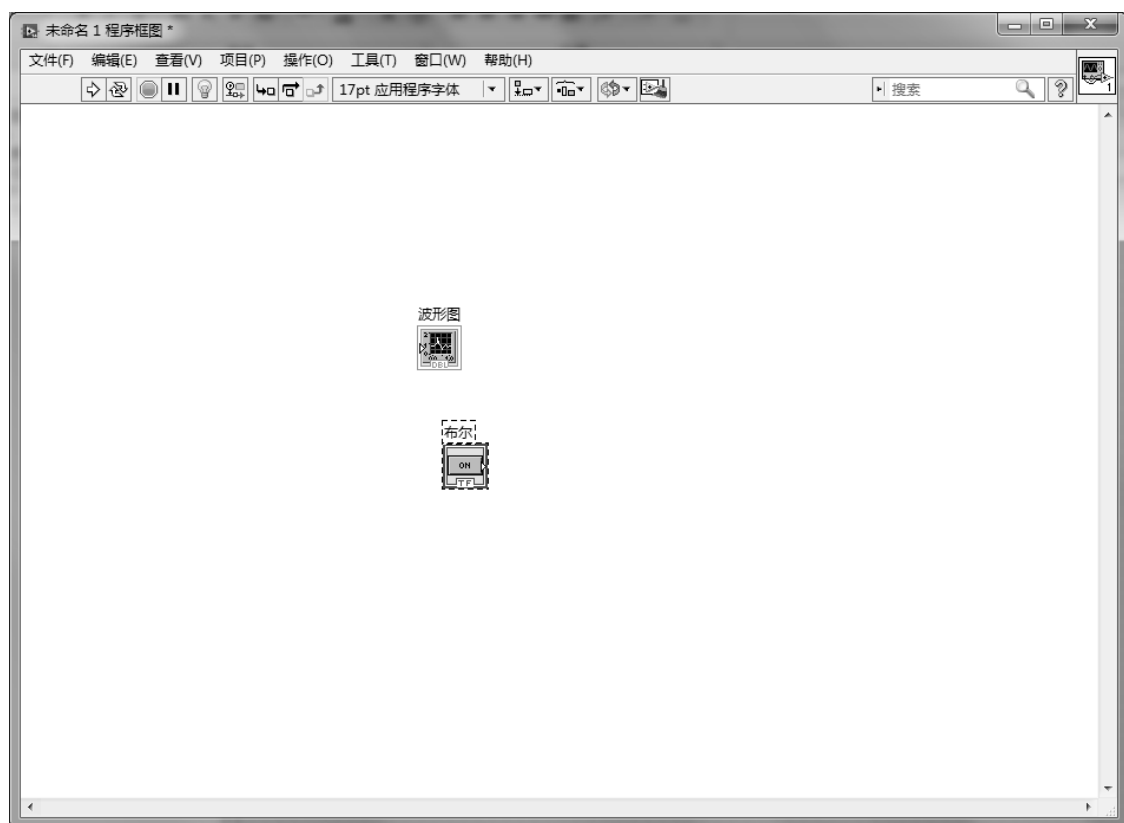


图 13.31 程序框图



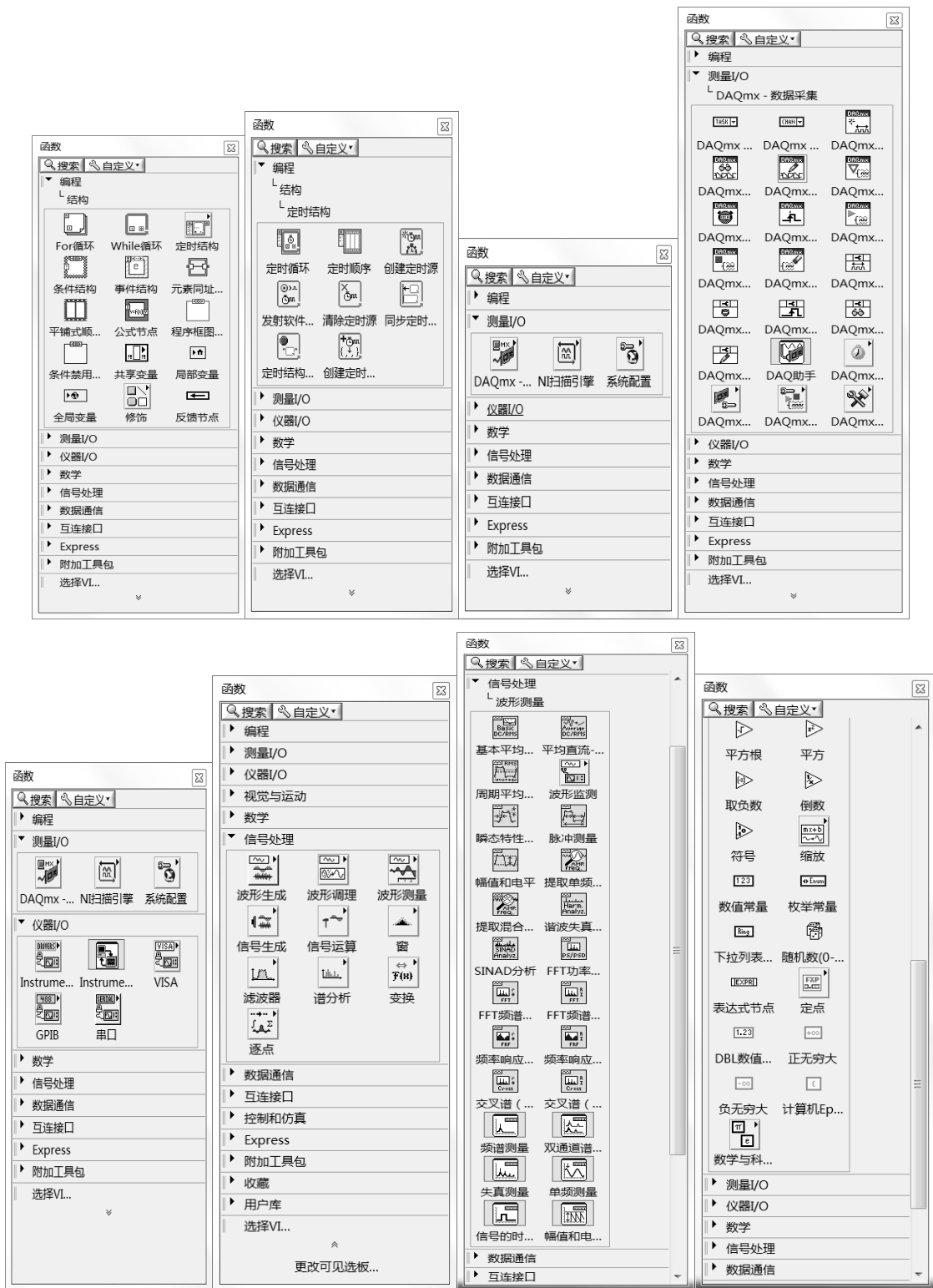


图 13.32 程序框图及其控件



图 13.32 程序框图及其控件 (续)

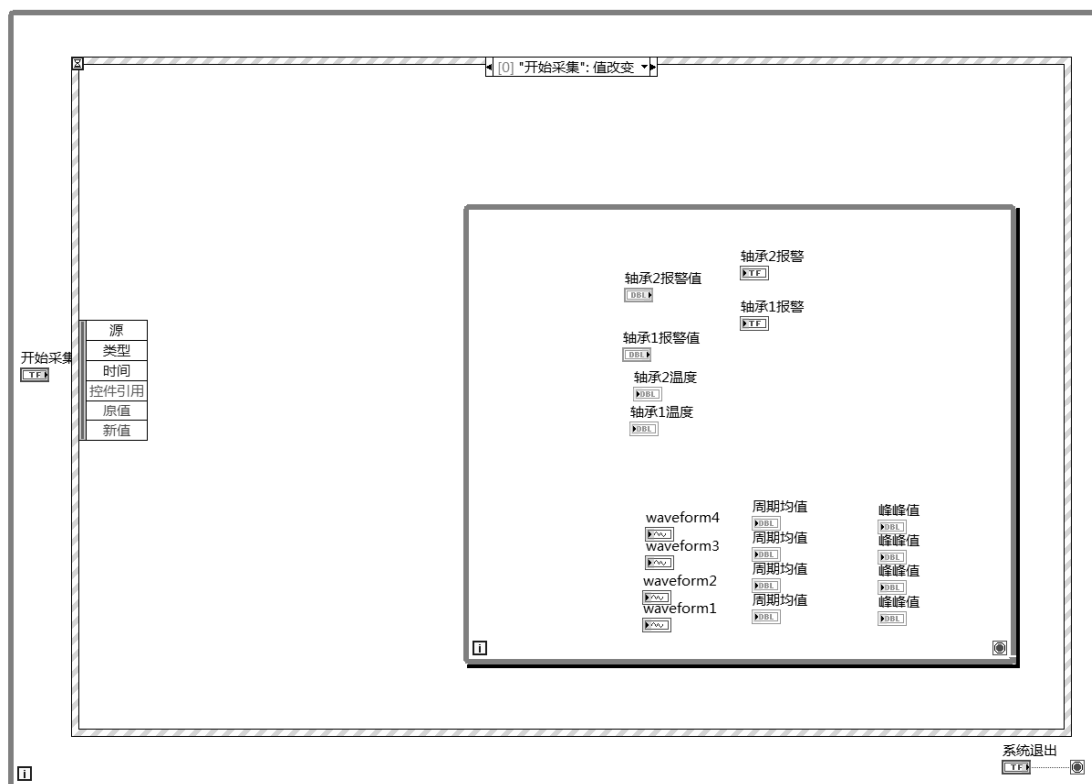


图 13.33 温度、振动结构控件

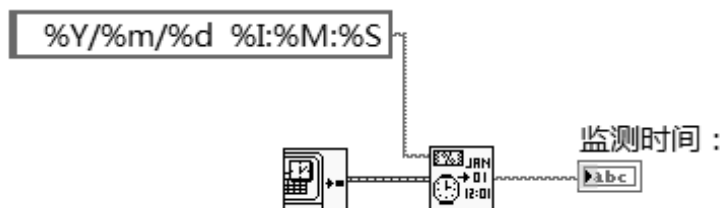


图 13.34 控件“监测时间”连线

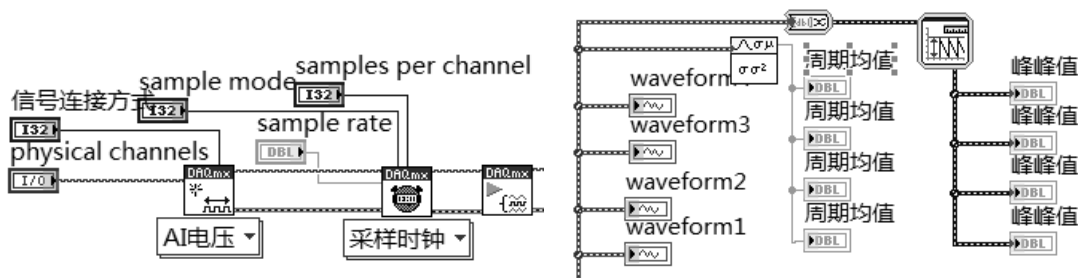


图 13.35 振动信号采集模块各控件的设置与连线

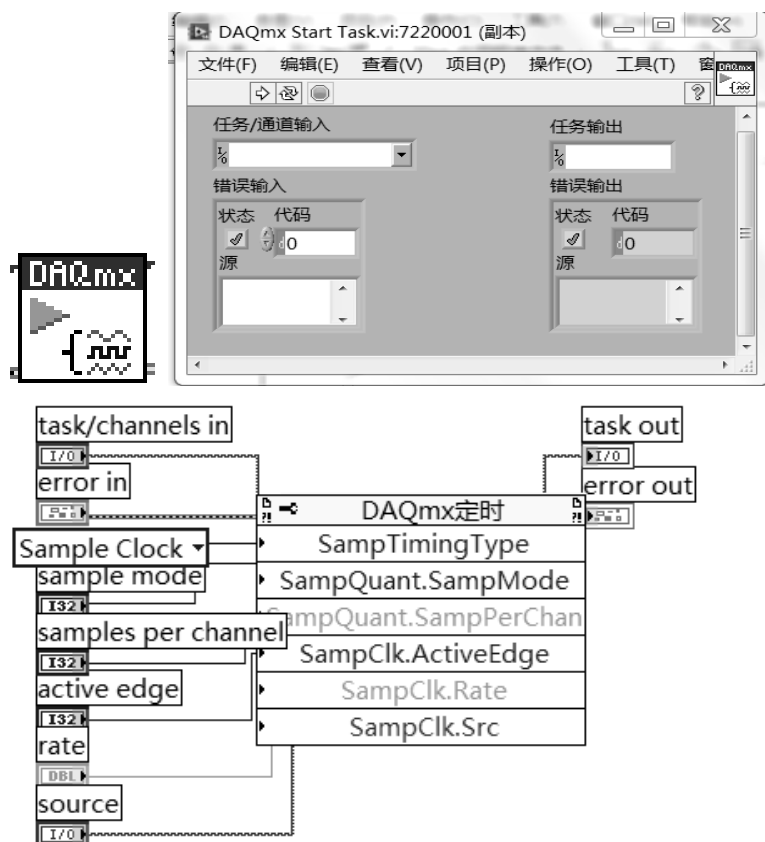


图 13.36 DAQmx 通道采集汇总 VI

标准差与方差处理 VI 如图 13.37 所示。

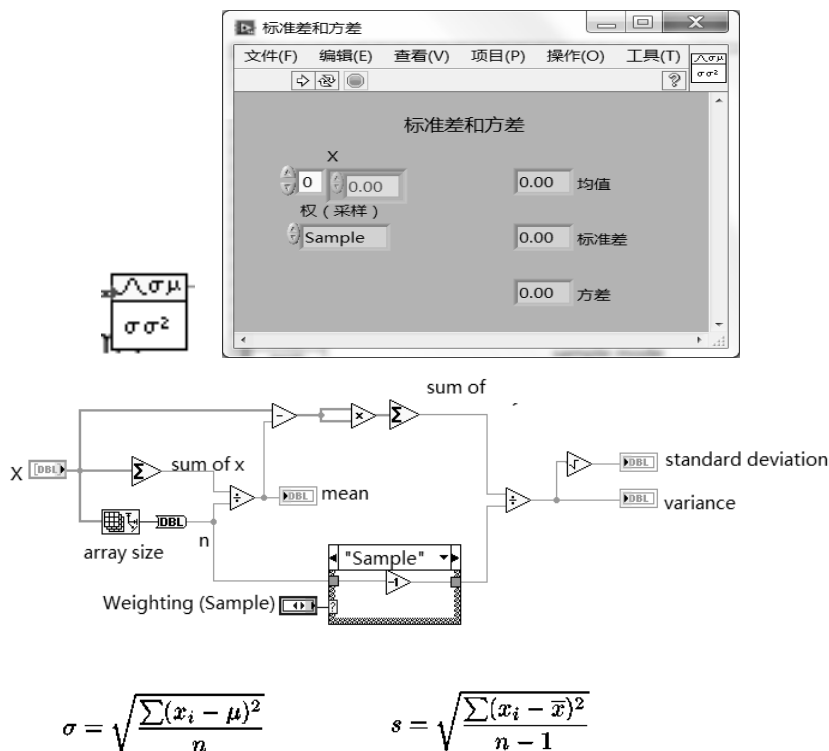


图 13.37 标准差与方差处理 VI

配置幅值和电平测量属性设置，如图 13.38 所示。

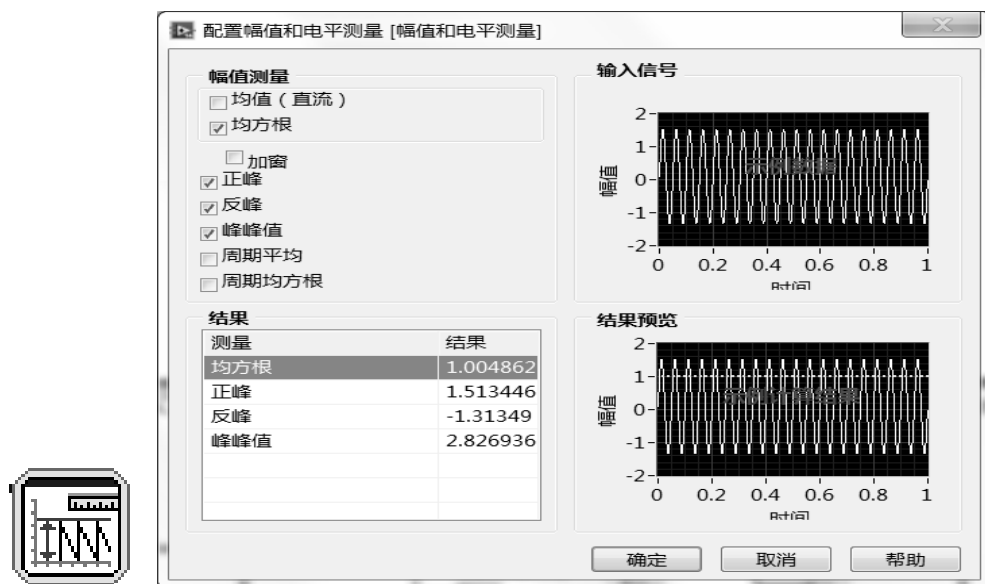


图 13.38 幅值和电平测量属性设置

错误处理器 VI 如图 13.39 所示。

错误输出汇总 VI 如图 13.40 所示。

(7) 连接完整的振动信号采集模块, 如图 13.41 所示。

矿机通风机轴承振动、温度在线监测系统最终的程序框图如图 13.42 所示, 系统启动后, 采集的工作状态下轴承振动、温度数据如图 13.43 所示。



图 13.39 错误处理器 VI



图 13.40 错误输出汇总 VI

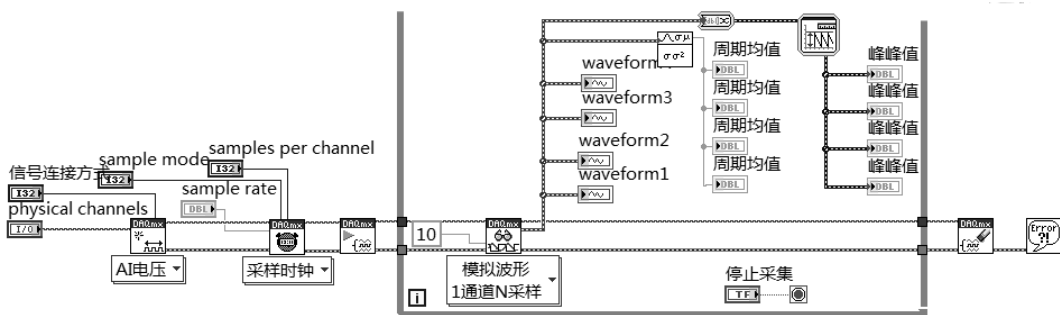


图 13.41 振动信号采集模块连接

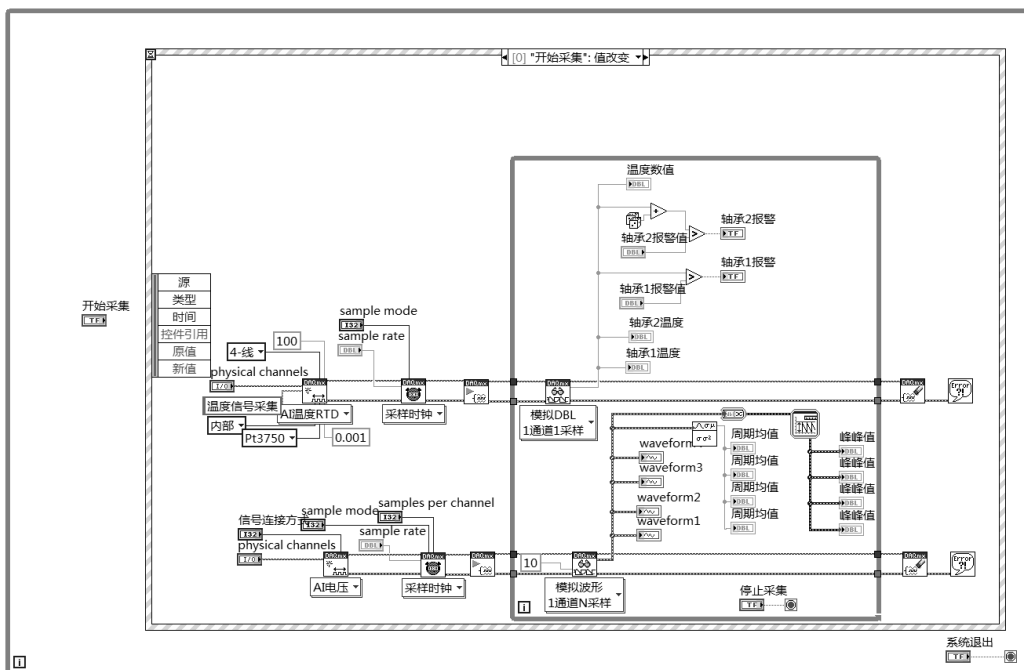


图 13.42 矿机通风机轴承振动、温度在线监测系统的程序框图

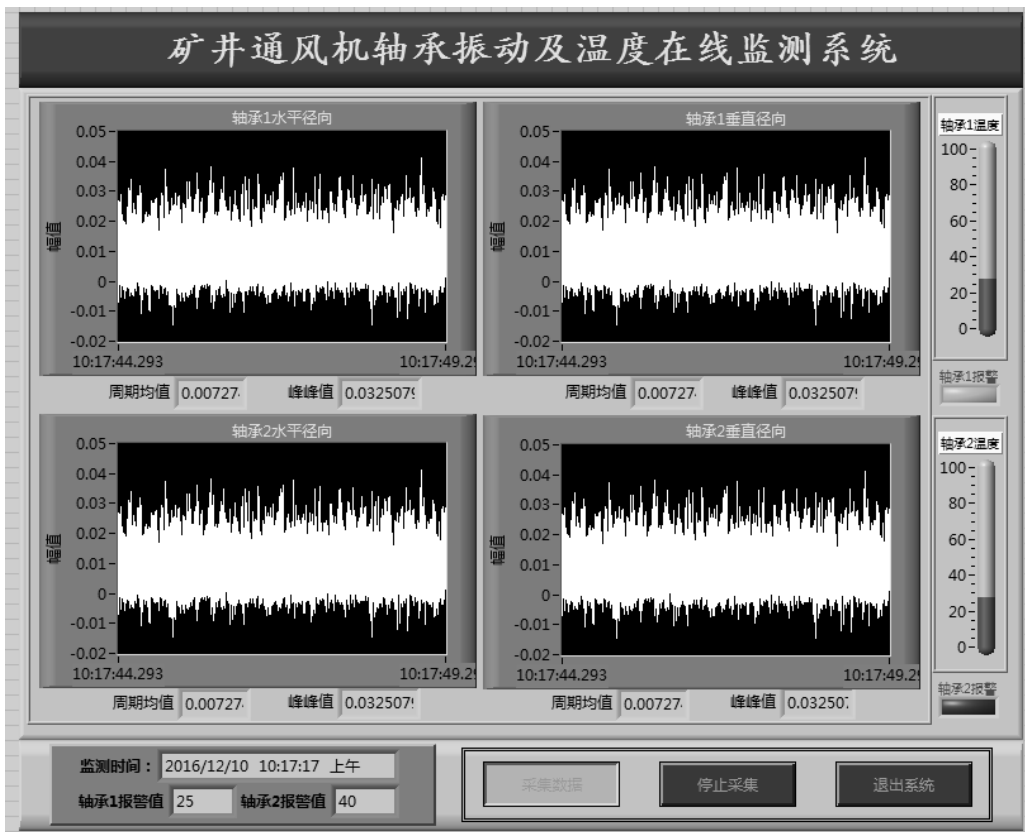
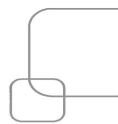


图 13.43 工作状态下轴承振动、温度数据



## 13.4 温度检测系统设计

温度是人们日常生产、生活中最常见的检测参数之一，温度不仅在生活中与我们时刻相关，而且在工厂生产、技术科研中也是一个非常值得关注的指数，每个物体发生地物理现象与化学性质的改变都会伴随温度变化，很多工厂的生产过程要把温度控制在一定范围之内，在相应的点对温度的检测与控制是非常常见的。

温度检测过程就是通过温度传感器将外界的物理信号转变成电流或电压信号，再经过数据采集卡将获得的电信号转换为数字信号输入至个人计算机中，从而完成数值分析、处理与显示。

### 13.4.1 设计目的

基于 LabVIEW 虚拟开发平台使用图形语言，利用软件对温度数据进行采集测量。在数据采集过程中温度计显示控件能够实时显示温度值。当采集过程结束后，在波形图上会显示温度波形。

本系统的设计内容主要包括温度采集部分，温度显示部分。本实例使用 LabVIEW 函数选板中的主要相关函数完成。

### 13.4.2 系统的硬件构成

系统硬件部分由温度传感器、信号调理电路、数据采集卡、计算机几部分组成。温度信号的处理过程：传感器把温度转换成电信号，经过调理电路，将信号放大、滤波，处理成可被数据采集卡采集的标准电压信号，在数据采集卡内将模拟信号转化为数字信号，并在数据采集指令下将其送入计算机总线，再在 PC 内进行各种仪器所需的功能处理。在数据采集的硬件实现过程中，主要包括两部分内容：传感器选型，数据采集设备的选择和配置。

#### 1) 传感器的选择

在温度测量中，最常用的是热电阻和热电偶。热电阻是将温度变化转换为电阻值的变化，而热电偶是将温度变化转换为电势变化。这两种温度传感器目前在工业生产中得到了广泛应用，本系统中选用热电阻测温度。

#### 2) 采集卡的选择

数据采集卡的功能是把模拟信号转换为数字信号，并把采集到的信号传输给计算机。计算机通过它来采集和处理模拟信号，它是数据采集硬件的核心。

本系统采用的 NI 9217 是 4 通道、24 位分辨率的电阻温度探测器 (RTD) 模拟输入模块，可进行  $100\Omega$  RTD 测量。NI 9217 可配置成两种不同的采样率模式。借助高采样率模式，采样率可达 400 Hz (每通道 100 Hz)。在高分辨率模式下，采样率为 5 Hz (每通道 1.25

Hz) 并具有 50/60 Hz 内置式去噪功能。NI 9217 可与 3 线和 4 线 RTD 测量兼容, 自动探测与通道连接的 RTD 类型 (3 线或者 4 线), 并可自动把每条通道配置成恰当的模式。该模块可提供每通道 1 mA 的电流激励, 其整个操作温度范围内的精度误差小于 1°C。NI 9217 包含可溯源至 NIST 的校准, 并具有通道-地面接地双重隔离屏障, 实现了安全性、抗扰性和高共模电压范围。

### 3) 系统前面板的设计

本例所设计的温度检测控制系统可对不同温度数据进行采集、检测, 并可以设置传感器类型及参数, 将采集到的温度数据显示在波形图表中。

系统前面板控件主要包括系统运行开关、波形图显示、通道设置、定时设置, 温度传感器参数等, 如图 13.44 所示。

前面板设计步骤:

(1) 在启动主界面的“新建”项中创建新 VI, 命名为“温度检测系统”。

(2) 通道设置。

a. 执行“控件→银色→I/O→DAQmx 名称控件→DAQmx 物理通道”操作, 添加 1 个物理通道控件。

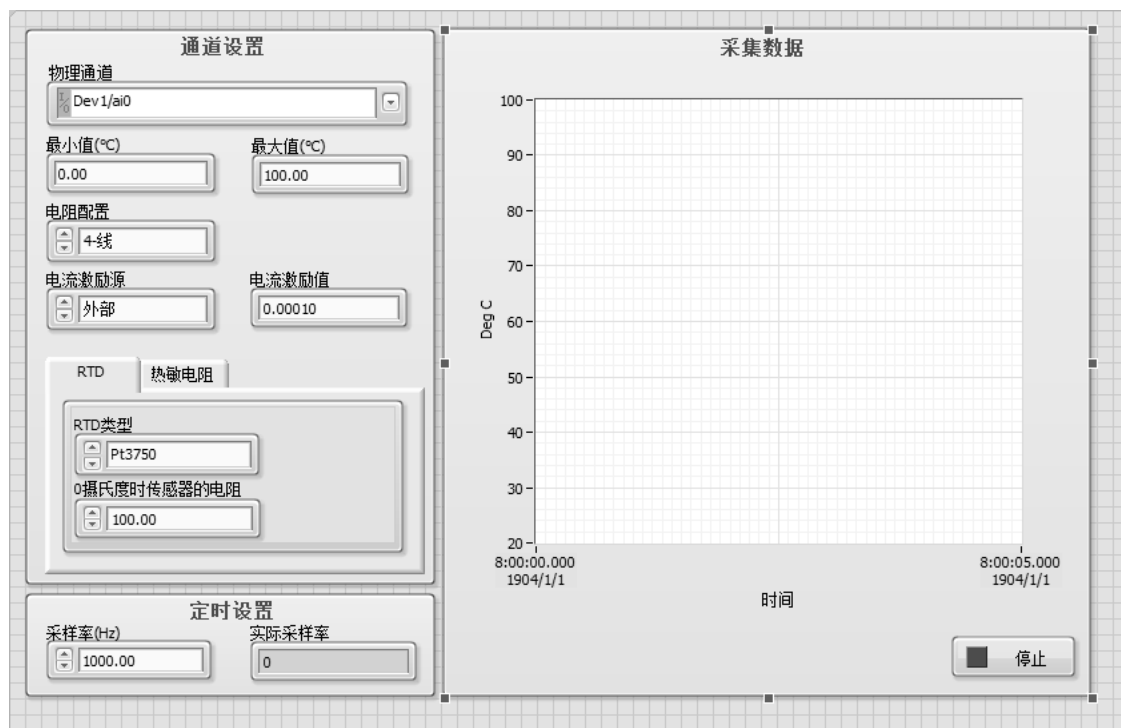


图 13.44 温度检测系统前面板

b. 执行“控件→银色→数值→数值输入控件”操作, 添加 3 个数值输入控件, 分别命名为“最大值”, “最小值”, “电流激励值”。

c. 执行“控件→银色→下拉列表与枚举→枚举”操作, 添加 2 个枚举, 分别命名为“电



阻配置”和“电流激励源”，在“电阻配置”中添加 3 项，分别为“2-线”、“3-线”、“4-线”，在“电流激励源”中添加 2 项，分别为“内部”、“外部”。设计界面如图 13.45 所示。



图 13.45 通道设置板块

(3) 传感器及参数的选择。

a. 执行“控件→银色→下拉列表与枚举—枚举”操作，添加 1 个枚举，命名为“RTD 类型”。在“RTD 类型”中添加 7 项，分别为“自定义”、“Pt3928”、“Pt3920”、“Pt3916”、“Pt3911”、“Pt3851”、“Pt3750”。

b. 执行“控件→银色→数值→数值输入控件”操作，添加 1 个数值输入控件，命名为“0 摄氏度时传感器的电阻”。

c. 执行“控件→银色→数组、矩阵与簇→簇”，添加一个簇，并将“RTD 类型”和“0 摄氏度时传感器的电阻”放入簇中，结果如图 13.46 所示。

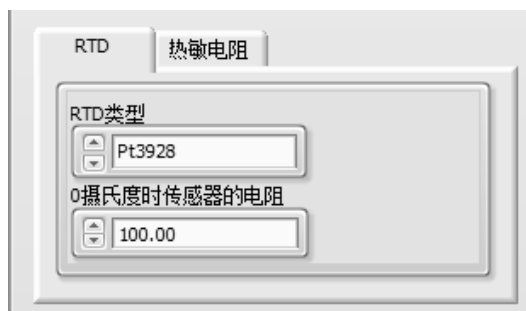


图 13.46 温度传感器及参数选择板块

(4) 定时设置。

执行“控件→银色→数值→数值输入控件”操作，添加 1 个数值输入控件，命名为“采样频率”。执行“控件→银色→数值→数值显示控件”操作，添加 1 个显示控件，命名为“实际采样率”，结果如图 13.47 所示。

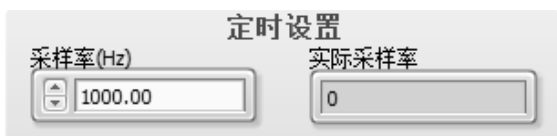


图 13.47 定时设置板块

(5) 采集数据显示。

执行“控件→银色→图形→波形图表”操作，添加一个显示波形的控件，命名为“采集数据”，结果如图 13.48 所示。

#### 4) 程序框图的设计

程序设计的过程中使用了条件结构和 While 循环结构，这些程序的使用是确保温度检测系统能够在便捷、快速、有利于操作者使用的环境下顺利执行和完成实验任务。

(1) 执行“函数→编程→结构→条件结构”操作，添加一个条件结构，在“默认”中添加“RTD”。

(2) 执行“函数→测量 I/O→数据采集→DAQmx 创建虚拟通道”操作，添加一个“创建虚拟通道”函数，然后右键单击该图标处“AI 电压”，选择“模拟输入”→“温度”→“RTD”，此时就转换成测量温度的虚拟通道，并把它放入条件结构的“RTD”中，如图 13.49 所示。

(3) 执行“函数→编程→簇、类与变体→按名称解除捆绑”操作，添加按名称解除捆绑函数，并放入条件结构的“RTD”中。

(4) 将“最大值”、“最小值”、“电阻配置”、“电流激励值”、“物理通道”、“电流激励源”端子依次与创建虚拟通道函数的相应端口相连接，如图 13.50 所示。

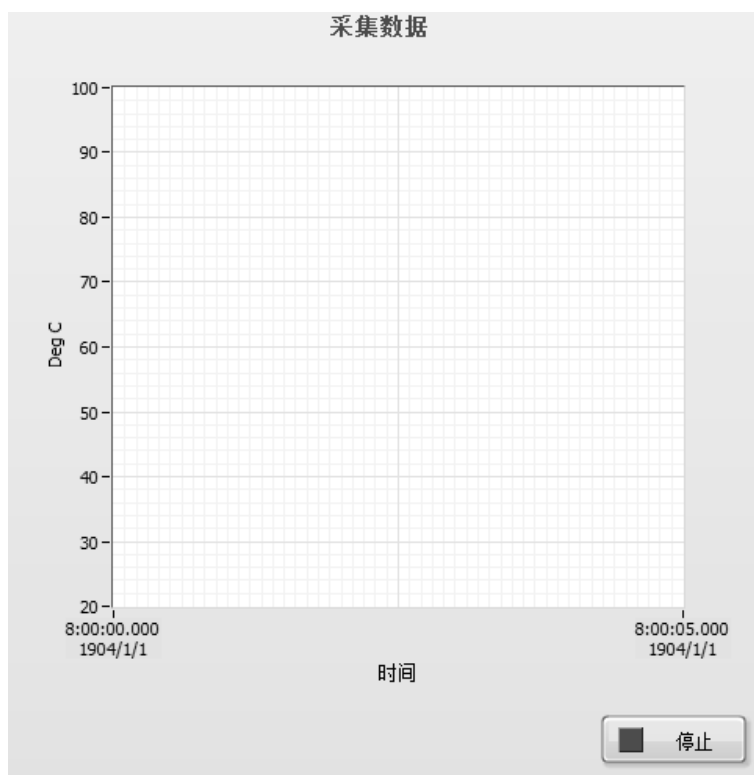


图 13.48 图形显示板块



图 13.49 创建虚拟通道

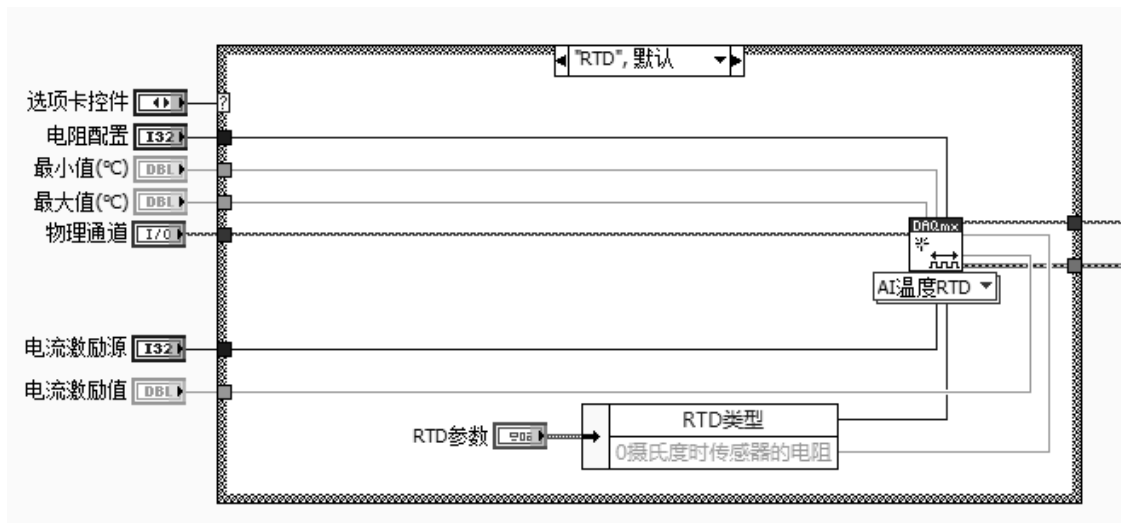


图 13.50 创建虚拟通道

(5) 执行“函数→测量 I/O→数据采集→DAQmx 定时”操作，添加一个定时函数。在“采样模式”端口创建常量，选择“连续采样”，将“采样率”端子与定时函数的“速率”端口相连，如图 13.51 所示。

(6) 执行“函数→编程→应用程序控制→属性节点”操作，添加一个属性节点“DAQmx 定时”，右键单击图标，出现菜单，单击“选择类”→“DAQmx”→“DAQmx 定时”，右键单击图标“属性”，单击“选择属性”→“采样时钟”→“速率”，完成属性节点的设置，如图 13.52 和图 13.53 所示。将“采样时钟”的输出端与“DAQmx 定时”的相应输入端相连，将“实际采样频率”与属性节点“DAQmx 定时”的“采样时钟：速率”端口相连，如图 13.54 所示。

(7) 执行“函数→测量 I/O→数据采集→DAQmx 开始任务”操作，添加一个开始任务函数，将“DAQmx 定时”的输出端和“DAQmx 开始任务”的相应输入端相连。

(8) 执行“函数→Express→执行过程控制→While 循环”操作，添加一个 While 循环结构。

(9) 在循环结构中添加“DAQmx 读取”函数，路径选择“函数→测量 I/O→数据采集→DAQmx 读取”，单击“模拟 DBL1 通道 1 采样”按钮，再选择菜单上“模拟”→“多通道”→“多采样”→“1D 波形”命令，在“超时”端口创建常量为 10，再将“数据”端口与“波形图表”端子相连，如图 13.55 所示。

(10) 执行“函数→编程→数值→转换→转换为长整型”操作，添加一个“转换为长整型”节点，放置在循环结构外，将“采样率”通过“转换为长整型”转换后与“DAQmx 读取”的“每通道采样数”端口相连。

(11) 执行“函数→测量 I/O→数据采集→DAQmx 停止任务”操作，添加一个“DAQmx 停止任务”函数，放置在循环结构外右侧。执行“函数→测量 I/O→数据采集→DAQmx 停止任务”操作，再添加一个“DAQmx 清除任务”函数。

(12) 执行“函数→编程→对话框与用户界面→通用错误处理器”操作，添加一个通用错误处理器，在“对话框类型”端口创建常量，改为“确定信息+警告”。

(13) 将“DAQmx 开始任务”、“DAQmx 读取”、“DAQmx 停止任务”、“DAQmx 清除任务”和“通用错误处理器”的相应端口相连。

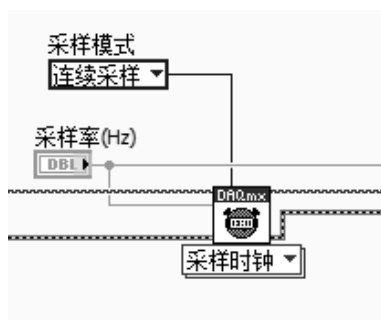


图 13.51 采样时钟



图 13.52 添加 DAQmx 定时



图 13.53 DAQmx 定时属性设置

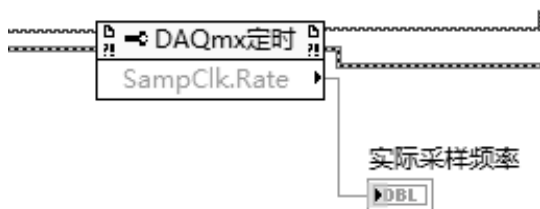


图 13.54 DAQmx 定时

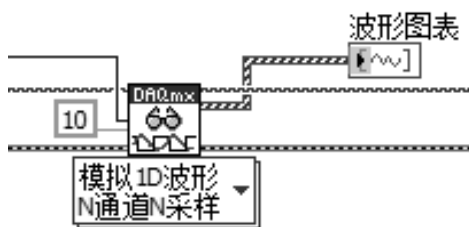


图 13.55 DAQmx 读取

(14) 在循环结构中执行“函数→编程→簇、类与变体→按名称解除捆绑”操作，添加一个按名称解除捆绑函数，鼠标单击“按名称解除捆绑”控件左键，选择 `status`。执行“函数→编程→布尔→或”操作，添加“或”操作函数。

(15) 将“DAQmx 读取”控件上的“错误输出”连接“按名称解除捆绑”函数，“按名称解除捆绑”和“停止”端子与“或”端子进行“或”运算，“或”运算结果连接 While 循环的终止条件，实现循环的控制，如图 13.56 所示。

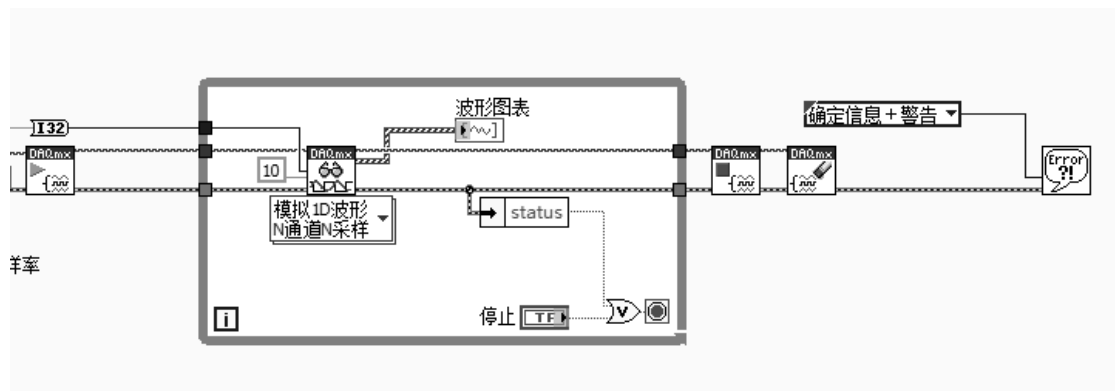


图 13.56 while 循环

最后得到程序框图如图 13.57 所示。

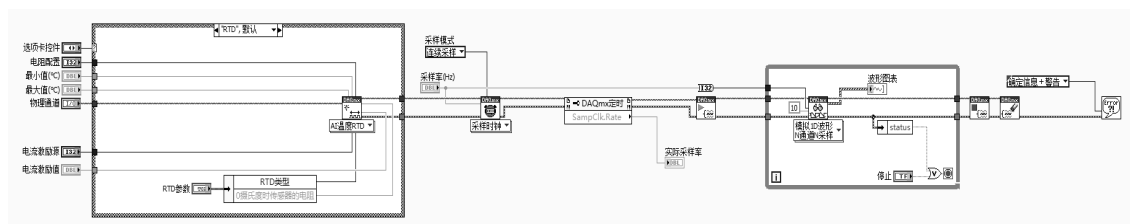


图 13.57 程序框图

## 13.5 压力测试系统

### 13.5.1 设计目的

本系统是利用虚拟仪器软件设计对液压系统中的压力参数进行测试，通过 LabVIEW 软件进行编程，基于软件开发环境，对数据采集系统中的信号进行传输、处理和显示，实现对压力实验参数的检测、分析和显示，实现用户自定义的试验系统。本系统可以使用仿真信号进行虚拟实验，也可以利用真实硬件进行数据采集、显示与分析。

### 13.5.2 系统前面板设计

首先在操作面板上选出所需要的指示控件和控制部件，并对操作面板的布局进行构思，使得操作面板更加合理和逼真。接下来要考虑的就是框图程序编程的步骤：在编程的过程中需要选择一些功能性模块来构建框图程序的结构，这样就能够编写出高效的程序了，也能够实现虚拟仪器面板上的各种功能了。LabVIEW 是一种图形化的编程语言，并且也提供了图形化的功能模块，合理的构思和组织自己所需的功能模块能够高效地完成实验中的各种测

量、显示以及保存等功能。

在前面板设计中包括控制按钮、参数的设定、数据的保存、数据的显示与分析以及报警功能，如图 13.58 所示。

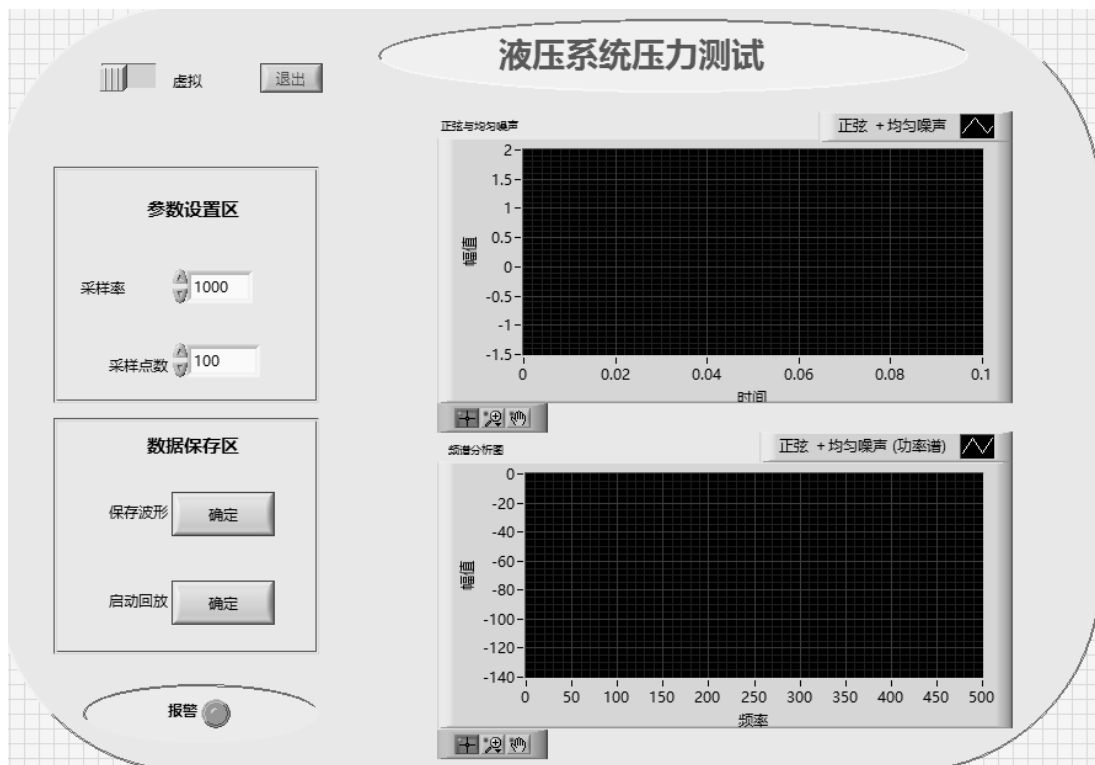


图 13.58 系统前面板

前面板设计具体实现步骤：

(1) 在启动主界面的“新建”项中创建新 VI，命名为压力测试系统。

(2) 控制按钮的设计。

a. 执行“控件→新式→经典→经典布尔→水平滑动开关”操作，添加一个水平滑动开关。

b. 执行“控件→新式→布尔→停止按钮”操作，添加一个停止按钮。

(3) 参数配置区的功能是设置测试参数，可以设置数据采集卡的通道、扫描速率与每个通道的采样点数。

执行“控件→新式→数值→数值输入控件”操作，添加两个数值输入控件，分别命名为“采样率”和“采样点数”，如图 13.59 所示。

(4) 数据保存区的功能主要是实现对数据的保存与数据回放。

执行“控件→新式→布尔→确定按钮”操作，添加二个确定按钮，分别命名为“保存波形”和“启动回放”，如图 13.60 所示。

(5) 报警功能设置。

执行“控件→新式→经典→经典布尔→圆形指示灯”操作，添加一个圆形指示灯显示控件并命名为“报警”。鼠标单击“圆形指示灯”，在右键下拉菜单中选择属性，出现对话框，

在外观一栏将“关”颜色设置为绿色。



图 13.59 参数设置

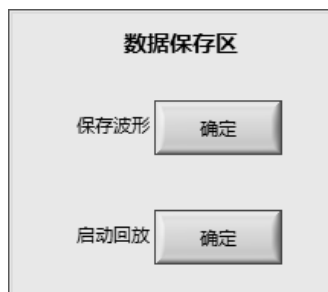


图 13.60 数据保存与回放

#### (6) 数据的显示与分析。

执行“控件→新式→图形→波形图”操作，添加两个波形图显示控件，分别命名为“正弦与均匀噪声”和“频谱分析图”，如图 13.61 所示。

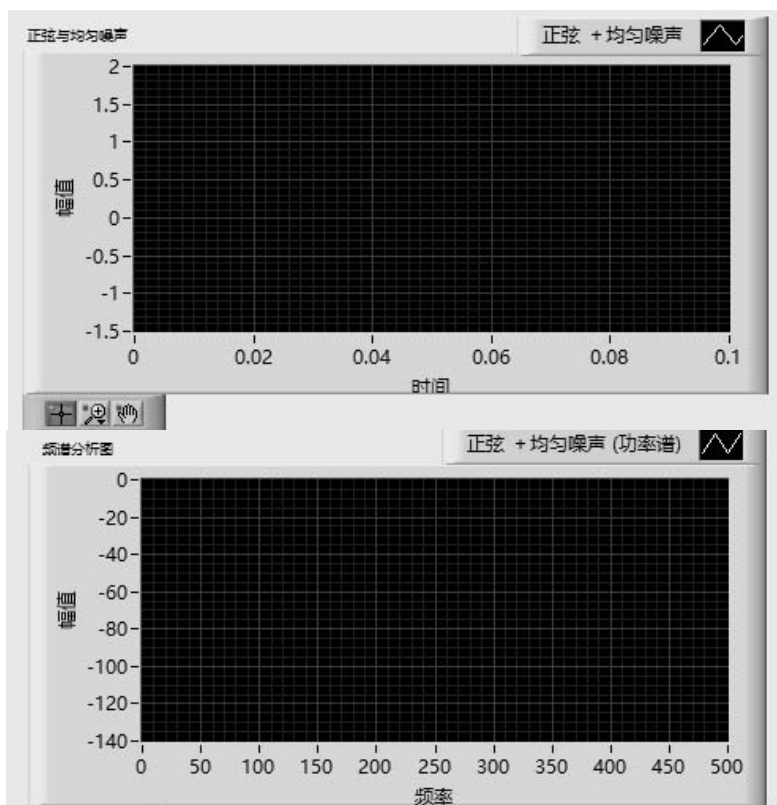


图 13.61 数据显示与分析

### 13.5.3 系统程序框图设计

压力测试控制的框图程序主要包括以下几个：数据产生/采集程序，数据保存程序，数据回放程序，数据显示与分析程序，报警程序，退出程序。



具体实现步骤:

(1) 执行“函数→编程→结构→While 循环”操作, 添加一个 While 循环。

(2) 数据产生/采集程序设计。

执行“函数→编程→结构→条件结构”操作, 添加一个条件结构。

鼠标单击“虚拟”控件, 在右键下拉菜单中选择“显示为图标”; 将“虚拟”控件连接到“条件结构”的选择器接线端。

当条件选择器标签为“假”时, 在条件结构中添加一个仿真信号发生器, 执行路径“函数→信号处理→波形生成→仿真信号”。

当条件选择器标签为“真”, 添加一个 DAQ 助手作为信号采集端, 执行路径“函数→编程→测量 I/O→DAQmx-数据采集→DAQ 助手”。鼠标单击“采样点数”控件, 在右键下拉菜单中选择“显示为图标”; 鼠标单击“采样率”控件, 在右键下拉菜单中选择“显示为图标”, 将“采样点数”和“采样率”连接到 DAQ 助手的采样点数和采样率端, 如图 13.62 所示。

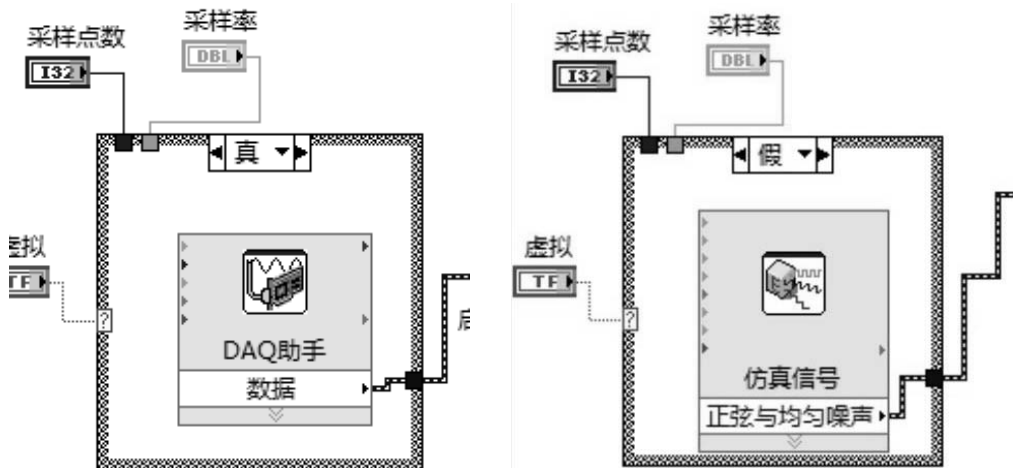


图 13.62 数据产生/采集

(3) 数据保存程序设计

a. 执行“函数→编程→结构→条件结构”操作, 添加一个条件结构。

b. 鼠标单击“保存波形”, 在右键下拉菜单中选择“显示为图标”, 将“保存波形”端子连接条件结构的选择器接线端。

c. 当条件选择器标签为“真”时, 执行“函数→编程→数据通信→局部变量”, 创建一个局部变量。鼠标单击左键, 出现下拉菜单, 选择“保存波形”。

执行“函数→编程→布尔→假常量”, 添加一个布尔型假常量, 将“假常量”连接到局部变量“保存波形”。

鼠标单击“启动回放”, 右键单击“创建→属性节点”, 选择“禁用”。

执行“函数→编程→数值→数值常量”, 创建一个数值常量并赋值 2, “数值常量”连接禁用“启动回放”。

执行“函数→编程→文件 I/O→写入测量文件”, 如图 13.63 所示。

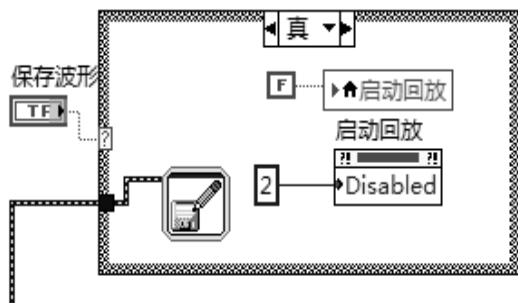


图 13.63 数据保存

d. 当条件选择器标签为“假”时，鼠标再次单击“启动回放”，右键单击“创建→属性节点”，选择“禁用”。

执行“函数→编程→数值→数值常量”，创建一数值常量并赋值 0，将数值常量与禁用“启动回放”连接，如图 13.64 所示。

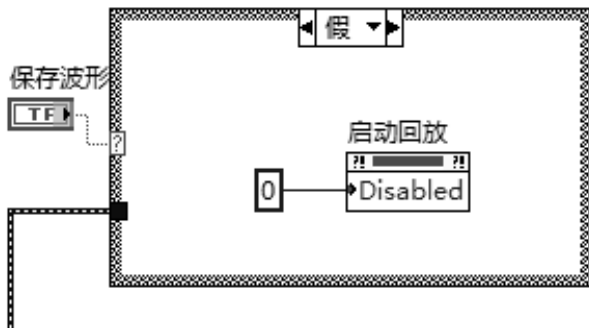


图 13.64 数据保存

#### (4) 数据回放程序设计。

a. 执行“函数→编程→结构→条件结构”，添加一个条件结构。

b. 鼠标单击“启动回放”，在右键下拉菜单中选择“显示为图标”，将“启动回放”连接到选择结构的选择器接线端。

c. 当条件选择器标签为“真”时，鼠标单击“保存波形”，在右键下拉菜单中选择“创建”，单击“属性节点”，选择“禁用”。鼠标单击该“禁用”属性节点，在右键下拉菜单中选择“转换为写入”，后单击该菜单“创建”，选择“常量”，鼠标单击该常量，左键选择“Disabled and Grayed Out”。

执行“函数→编程→数据通信→局部变量”，创建一个局部变量并命名为“保存波形”。执行“函数→编程→布尔→假常量”，创建一个布尔型假常量，将假常量连接到局部变量的“保存波形”。

执行“函数→编程→文件 I/O→读取测量文件”操作，创建一个读取测量文件节点。执行“函数→编程→对话框与用户界面——通用错误处理器”操作，创建一个通用错误处理器节点。将读取测量文件“错误输出”端口连接到通用错误处理器“错误输入（无错

误)”端口。

执行“函数→编程→数值→数值常量”操作，创建一个数值常量并赋值 4，将数值常量连接到通用错误处理器的错误代码端。

鼠标单击“通用错误处理器”对话框类型端口，在右键下拉菜单中选择“创建→常量”，然后鼠标单击该常量，左键选择“无对话框”，如图 13.65 所示。

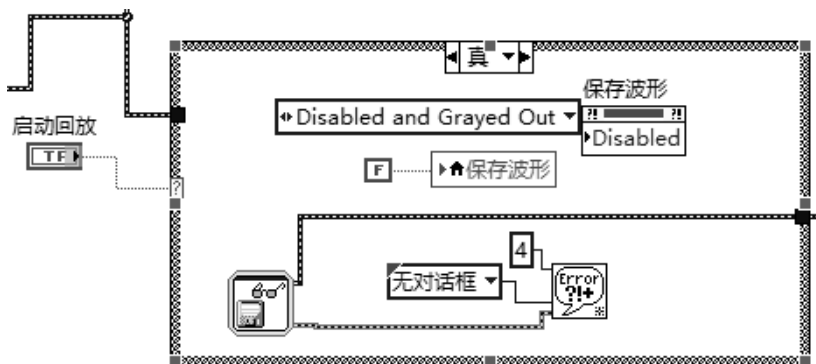


图 13.65 数据回放 1

d. 当条件选择器标签为“假”时，鼠标单击“保存波形”，在右键下拉菜单中选择“创建→属性节点→禁用”。

执行“函数→编程→数值→数值常量”操作，创建一个数值常量并赋值 0，将数值常量连接到属性节点“保存波形”如图 13.66 所示。

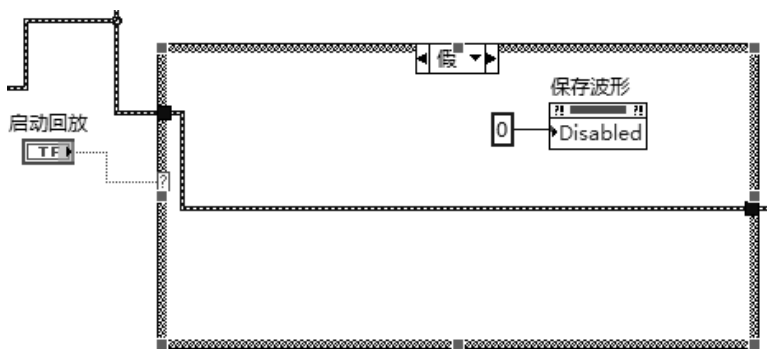


图 13.66 数据回放 2

(5) 数据产生/采集程序、数据存储以及数据显示程序的连接如图 13.67 所示。

(6) 数据显示与分析程序设计。

a. 将数据回放程序与“压力曲线”相连接。

b. 执行“函数→编程→信号处理→波形调理→滤波器”操作，创建一个滤波器函数。在数据回放程序与“压力曲线”连线上创建分支，连接滤波器的“信号”端口。

c. 执行“函数→编程→结构→信号处理→波形测量→频谱测量”操作，创建一个频谱测量函数。将滤波器的“滤波后的信号”端口连接到频谱测量的“信号”端口。频谱测量的

“功率谱”端口连接到“频谱分析图”端子，如图 13.68 所示。

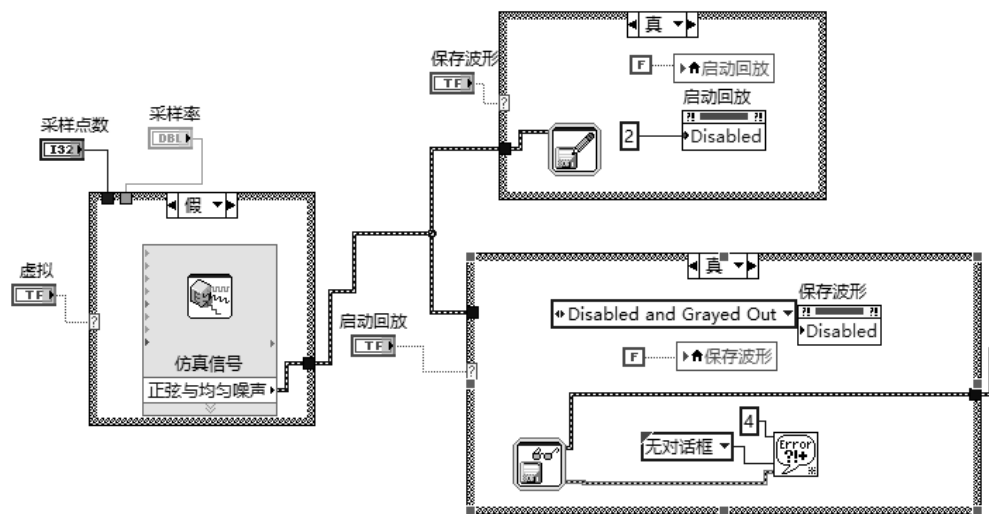


图 13.67 程序间的连线

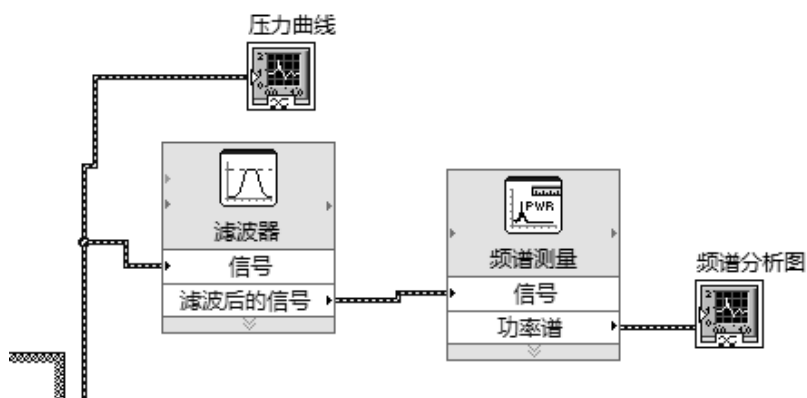


图 13.68 数据显示与分析

#### (7) 报警程序设计。

- 执行“函数→信号处理→波形测量→提取单频信息”操作，创建一个提取单频信息函数，将数据回放中的“读取测量文件”的信号端口连接到提取单频信息。
- 执行“函数→编程→比较→大于”操作，添加一个大于操作节点。  
执行“函数→编程→数值→数值常量”操作，创建一个数值常量并赋值 0.8。  
执行“函数→编程→比较→小于”操作，添加一个小于操作节点。  
执行“函数→编程→数值→数值常量”操作，创建一个数值常量并赋值 3。  
执行“函数→编程→布尔→与”操作，创建一个操作节点。  
执行“函数→编程→数组→索引数组”，创建一个索引数组。
- 执行“函数→编程→结构→条件结构”操作，添加一个条件结构，将索引数组连接到条件结构的选择器接线端。

当条件选择器标签为“真”，在结构内执行“函数→编程→布尔→假常量”操作，创建

一个布尔型假常量。

当条件选择器标签为“假”，在结构内执行“函数→编程→布尔→真常量”操作，创建一个布尔型真常量。

条件结构连接到“报警”端子，如图 13.69 所示。

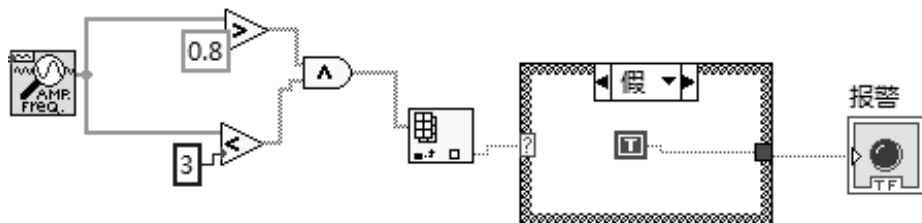


图 13.69 报警

(8) 鼠标单击“启动回放”，在右键下拉菜单中选择“创建→属性节点→禁用”，将该“禁用”放在 While 循环外，然后单击该“禁用”，右键选择“转换为写入→创建→常量”。

最终的程序框图如图 13.70 所示。

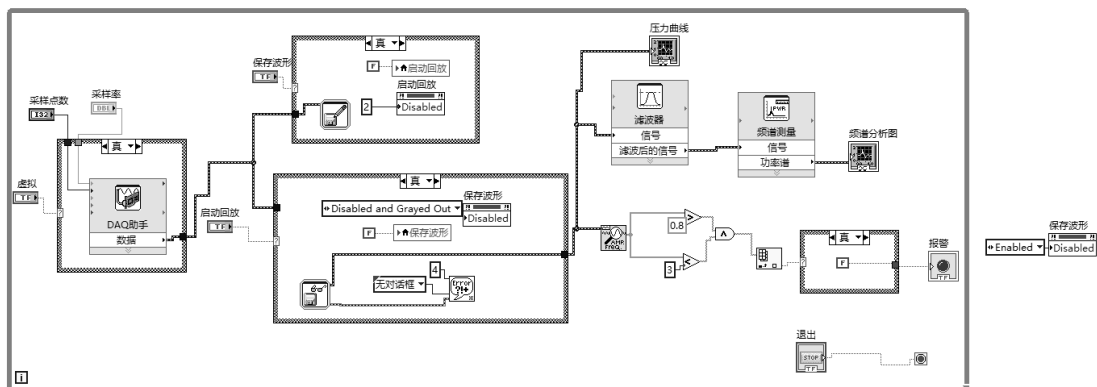


图 13.70 程序框图

# 附录 A LabVIEW 13.0 快捷键汇总

下面列出了 LabVIEW 环境下的快捷键。除以下快捷键外，用户也可以为 VI 菜单项创建自定义的快捷方式。

下列快捷方式中的<Ctrl>键对应于 OS X 的<Option>或<Command>键，Linux 的<Alt>键。

A-1 对象/动作快捷键

键盘组合	说 明
Shift+单击	选取多个对象；将对象添加到当前选择之中
方向箭头键	将选中的对象每次移动一个像素
Shift+方向箭头键	将选中的对象每次移动若干像素
Shift+单击（拖曳）	沿轴线移动对象
Ctrl+K	对象重新排序，将选中的对象在一组对象中前移一层
Ctrl+J	对象重新排序，将选中的对象在一组对象中后移一层
Ctrl+Shift+K	对象重新排序，将选中的对象移至一组对象的顶层
Ctrl+Shift+J	对象重新排序，将选中的对象移至一组对象的底层
Ctrl+单击（拖曳）	复制选中对象
Ctrl+Shift+单击（拖曳）	复制选中对象，并沿轴线移动对象
Shift+调整大小	调整选中对象的大小，并保持长宽比例
Ctrl+调整大小	调整选中对象的大小，并保持中心点
Ctrl+Shift+调整大小	调整选中对象的大小，并保持中心点和长宽比例
Ctrl+空白位置拖曳出长形区域	在前面板或程序框图上添加更多工作区间
Ctrl+A	选择前面板或程序框图上的所有对象
Ctrl+Shift+A	在对象上进行最终对齐操作
Ctrl+D	在对象上进行最终分布操作
双击空白区域	如已启用自动工具选择，将在前面板或程序框图上添加一个自由标签
Ctrl+鼠标滚轮	一次浏览条件、时间或层叠式顺序结构的子程序图
空白键（拖曳）	移动标签和标题时，禁用预设对齐位置
Ctrl+U	重新连接已有连线并重新自动排列程序框图对象

A-2 浏览 LabVIEW 环境快捷键

键盘组合	说 明
Ctrl+F	搜索对象或文本
Ctrl+Shift+F	打开搜索结果窗口或查找项目对话框，其中显示上次搜索结果
Ctrl+G	搜索 VI 中的下一个对象或文本
Ctrl+Shift+G	搜索 VI 中的上一个 VI、对象或文本
Ctrl+Tab	根据窗口在屏幕上显示的顺序依次浏览 LabVIEW 窗口。<Linux>窗口的顺序取决于窗口管理器



(续表)

键盘组合	说 明
Ctrl+Shift+Tab	按照相反的顺序浏览 LabVIEW 窗口
Ctrl+L	显示错误列表窗口
Ctrl+Shift+E	在项目浏览器窗口显示当前 VI
Ctrl+Shift+W	显示全部窗口对话框
Ctrl+Shift+B	打开类浏览器窗口

## A-3 浏览前面板和程序框图快捷键

键盘组合	说 明
Ctrl+E	显示前面板或程序框图
Ctrl+Space	显示快速放置对话框。在中文键盘上按<Ctrl+Shift+Space>键。<OS X>按<Command+Shift+Space>组合键
Ctrl+#	启用或禁用网格对齐。在法语键盘上，按<Ctrl+’>键。<OS X>按<Command+*>键
Ctrl+/	最大化窗口或还原窗口原始大小
Ctrl+T	分左右或上下两栏显示前面板和程序图
Ctrl+Shift+N	显示导航窗口
Ctrl+I	显示 VI 属性对话框
Ctrl+Y	显示历史窗口

## A-4 浏览 VI 层次结构窗口快捷键

键盘组合	说 明
Ctrl+D	重新绘制 VI 层次结构窗口
Ctrl+A	在 VI 层次结构窗口中显示所有 VI
Ctrl+单击 VI	显示在 VI 层次结构窗口中选中 VI 的子 VI 和其他构成 VI 的节点
Enter	在 VI 层次结构窗口中初始化一个搜索，输入文本，按回车键查找下一个匹配的节点
Shift+Enter	在 VI 层次结构窗口中初始化一个搜索，输入文本，按<Shift+Enter>键查找上一个匹配的节点

## A-5 调试快捷键

键盘组合	说 明
Ctrl+向下箭头	单步步入节点
Ctrl+向右箭头	单步步过节点
Ctrl+向上箭头	单步步出节点

## A-6 文件操作快捷键

键盘组合	说 明
Ctrl+N	打出一个空 VI
Ctrl+O	打出一个现有 VI



(续表)

键盘组合	说 明
Ctrl+W	关闭 VI
Ctrl+S	保存 VI
Ctrl+Shift+S	保存所有打开的文件
Ctrl+P	打印窗口
Ctrl+Q	退出 LabVIEW

## A-7 基本编辑快捷键

键盘组合	说 明
Ctrl+Z	撤销上次操作
Ctrl+Shift+Z	重做上次操作
Ctrl+X	剪切选中对象
Ctrl+C	复制选中对象
Ctrl+V	粘贴最近剪切或复制的对象

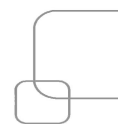
## A-8 帮助快捷键

键盘组合	说 明
Ctrl+H	显示即时帮助窗口。<OS X>按<Command+Shift+Space>组合键
Ctrl+Shift+L	锁定即时帮助窗口
Ctrl+? 或<F1>	显示 LabVIEW 帮助

## A-9 工具和选板快捷键

键盘组合	说 明
Ctrl	切换到下一个最有用的工具
Shift	切换至定位工具
在空白区域按 <Ctrl+Shift>	切换至滚动工具
空白键	如自动工具选择被禁用，在两个最常用工具间循环选择
Shift+Tab	启用自动工具选择
Tab	如通过单击自动工具选择按钮禁用了自动工具选择，按<Tab>键可在最常用的 4 个工具间循环选择。如通过其他方式禁用了自动工具选择，则按<Tab>键将启动自动工具选择
方向箭头键	在临时的控件和函数选板上进行方向移动
Enter	选择并进入一个临时选板
Esc	跳出一个临时选板
Shift+右键单击	在光标处显示临时的工具选板





## A-10 子 VI 快捷键

键盘组合	说 明
双击子 VI	显示子 VI 的前面板
Ctrl+双击子 VI	显示子 VI 的前面板和程序框图
拖曳 VI 图标至程序框图	将该 VI 作为子 VI 放置在程序框图上
Shift+拖曳 VI 图标至程序框图	将该 VI 作为子 VI 放置在程序框图上, 并将没有默认值的输入控件与常量相连
Ctrl+右键单击程序框图并从选板中选择 VI	打开所选 VI 的前面板

## A-11 执行快捷键

键盘组合	说 明
Ctrl+R	运行 VI
Ctrl+.	停止 VI, 在 VI 运行时使用
Ctrl+M	切换至运行或编辑模式
Ctrl+运行按钮	重新编译当前 VI
Ctrl+Shift+运行按钮	重新编译内存中的所有 VI
Ctrl+向下箭头	将选中光标移入数据组或簇, 在 VI 运行时使用
Ctrl+向上箭头	将选中光标移出数据组或簇, 在 VI 运行时使用
Tab	按 Tab 键顺序轮流选择控件, 在 VI 运行时使用
Shift+Tab	按 Tab 键反序选择控件, 在 VI 运行时使用

## A-12 连线快捷键

键盘组合	说 明
Ctrl+B	删除 VI 中的所有断线。如选择的结构或程序框图中有断线, 该快捷方式仅删除选中区域的断线
Esc, 右键单击或单击接线端	取消已开始的连线操作
单击连线	选中一个连线段
双击连线	选中一个连线分支
三击连线	选中整条连线
A	连线时, 暂停禁用自动连线路径选择
双击	连线时暂时停止另一端的连接, 不连往另一个接线端
空白键	将连线方向在水平和垂直方向之间选择
空白键	移动对象时, 在自动连线和自动连线之间切换
Shift+单击	撤销对连线设置的最后一个点
Ctrl+单击有两个输入端函数的某个输入端	互换两条输入连线的位置

## A-13 文本快捷键

键盘组合	说 明
双击	选中字符串中的一个单词
三次单击	选中整个字符串



(续表)

键盘组合	说 明
Ctrl+向右箭头	使用单字节字符的文本（如西方字符集）时，在字符串中向前移一个词。使用多字节字符的文本（如亚洲字符集）时，在字符串中向前移一个字符
Ctrl+向左箭头	使用单字节字符的文本（如西方字符集）时，在字符串中向后移一个词。使用多字节字符的文本（如亚洲字符集）时，在字符串中向后移一个字符
Home	移至字符串当前行的行首
End	移至字符串当前行的行尾
Ctrl+Home	移至整个字符串的开始位置
Ctrl+End	移至整个字符串的结束位置
Shift+Enter	在枚举型控件和常量、下拉列表控件和常量或条件结构中添加新项。在字符串常量中，按下<Shift+Enter>键，可禁用已启用的自动调整大小功能。如已禁用自动调整大小功能，按下<Shift+Enter>键时将在常量中显示滚动条
Ctrl+Shift+Enter	编辑条件选择器标签时复制条件结构的可见分支
Esc	取消当前对字符串的编辑
Ctrl+Enter	结束字符串输入
Ctrl+=	加大当前字号
Ctrl+-	减小当前字号
Ctrl+0	显示字体样式对话框
Ctrl+1	改变字体样式对话框中应用程序的字体
Ctrl+2	改变字体样式对话框中的系统字体
Ctrl+3	改变字体样式对话框中的对话框字体
Ctrl+4	改变字体样式对话框中的当前字体

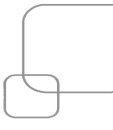
下面列出了快速放置对话框为活动状态时可以使用的键盘组合方式。在英文键盘上，按<Ctrl+Space>组合键打开快速放置对话框。中文键盘按<Ctrl+Shift+Space>组合键。（OS X）按<Command+Shift+Space>组合键。（Linux）按<Alt+Space>组合键。

此外，还可使用快速放置配置对话框创建自定义快速放置组合方式。如果要打开快速放置配置对话框，则需要单击快速放置对话框的配置按钮。

值得注意的是，如果要对指定的程序框图或前面板对象应用快捷方式，需在打开快速放置对话框前选中该对象。

A-14 快速放置快捷键

键盘组合	说 明
Ctrl+D	为所选程序框图对象所有未连接的输入和输出创建输入或显示控件
Ctrl+Shift+D	为所选程序框图对象所有未连接的输入创建常量
Ctrl+W	为选中的一排或平行的多排程序框图对象连线
Ctrl+Shift+W	为选中的一排或平行的多排程序框图对象连线，并整理所选对象连线
Ctrl+R	删除所选程序框图对象及与其相连的联系和常量，并为先前连接至该对象输入、输出端的相同数据类型连线
Ctrl+T	将 VI 顶层程序框图中所有输入控件的标签移至接线端左侧，显示控件的标签移至接线端右侧。如果只需要移动选中对象的标签设置，可选择程序框图上的多个对象，打开快速放置对话框，然后按<Ctrl+T>组合键



(续表)

键盘组合	说 明
Ctrl+Shift+T	将 VI 程序框图中所有输入控件（包括子程序框图中的输出控件）的标签移至接线端的左侧，显示控件的标签移至接线端的右侧
Ctrl+P	将所选前面板或程序框图对象替换为当前在快速放置对话框中选中的对象
Ctrl+I	在选中的程序框图连线上插入快速放置对话框中选中的对象
Ctrl+Shift+I	在多条选中的连线上插入快速放置对话框中所选对象的单个实例
Ctrl+B	将所选属性节点、调用节点和/或类说明符常量的 VI 服务器类更改为快速放置窗口中输入的类
Ctrl+Shift+B	将选中的属性节点或调用节点的属性或方法转换为快速放置对话框中输入的属性或方法

# 附录 B 公式节点和表达式节点 中的内建函数

B-1 内 建 函 数

函 数 名 称	对应 LabVIEW 图形函数	功 能 说 明
abs (x)	Absolute Value	返回 x 的绝对值
acos (x)	Inverse Cosine	计算 x 的反余弦
acosh (x)	Inverse Hyperbolic Cosine	计算 x 的反双曲余弦
asin (x)	Inverse Sine	计算 x 的正弦
asinh (x)	Inverse Hyperbolic Sine	计算 x 的反双曲正弦
atan (x)	Inverse Tangent	计算 x 的反正切
atan2 (y,x)	Inverse Tangent (2Input)	计算 y/x 的反正切
atanh (x)	Inverse Hyperbolic Tangent	计算 x 的反双曲正切
ceil (x)	Round To +Infinity	取整 x 到大于或等于 x 的最小整数
cos (x)	Cosine	计算 x 的余弦
cosh (x)	Hyperbolic cosine	计算 x 的双曲余弦
cot (x)	Cotangent	计算 x 的余切, 即 $1/\tan(x)$
csc (x)	Cosecant	计算 x 的余割, 即 $1/\sin(x)$
exp (x)	Exponential	计算 e 的 x 次幂
expm1 (x)	Exponential (Arg) -1	计算 e 的 x 次幂减 1
floor (x)	Round To -Infinity	取整 x 到小于或等于 x 的最大整数
getexp (x)	Mantissa&Exponent	返回 x 的指数
getman (x)	Mantissa&Exponent	返回 x 的尾数
Int (x)	Round To Nearest	四舍五入 x 到最接近的整数
Intrz (x)	--	四舍五入 x 到 x 和 0 最接近的整数
In (x)	Natural Logarithm	计算 x 的自然对数
Inp1 (x)	Natural Logarithm (Arg+1)	计算 x+1 的自然对数
log (x)	Logarithm Base 10	计算 x 以 10 为底的对数
log2 (x)	Logarithm Base 2	计算 x 以 2 为底的对数
max (x,y)	Max&Min	比较 x 和 y 并返回较大者的值
min (x,y)	Max&Min	比较 x 和 y 并返回较小者的值
mod (x,y)	Quotient & Remainder	计算 x/y 的余数, 商向负无穷四舍五入
pow (x,y)	Power of X	计算 x 的 y 次幂
rand ()	Random NUMBER (0-1)	生成 0~1 之间的随机浮点数
rem (x,y)	Quotient&Remainder	计算 x/y 的余数, 商向最接近整数四舍五入
sec (x)	Secant	计算 x 的正割, 即 $1/\cos(x)$



(续表)

函 数 名 称	对应 LabVIEW 图形函数	功 能 说 明
sign (x)	Sign	符号函数，若 x 为整数返回 1，若 x 为负数返回-1，若 x 为 0
sin (x)	Sine	计算 x 的正弦
Sinc (x)	sinc	计算 x 的正弦除以 x 的值
Sinh (x)	Hyperbolic	计算 x 的双曲正弦
sizeofDim (ary,di)	--	返回数组 ary 的第 di 维的大小
sqrt (x)	Square Root	计算 x 的平方根
tan (x)	Tangent	计算 x 的正切
tanh (x)	Hyperbolic Tangent	计算 x 的双曲正切

注：在表中的函数参数或返回结果中，角度值均以弧度为单位。

## 附录 C 公式节点和表达式节点 中的数学运算符

运算符	功能说明
**	求幂
+, -, !, ~, ++, —	均为一元运算符，功能依次是： 正、负、逻辑反、按位求反、前置或后置增一、前置或后置减一、++和—在表达式节点中可不用
*, /, %	二元运算符，乘、除、求余
+, -	二元运算符，加、减
>>, <<	算术右移或左移
>, <, >=, <=	大于、小于、大于或等于、小于或等于
!=, ==	逻辑运算符，不相等和相等
atanh(x)	计算 x 的反双曲正切
&	按位与
^	按位异或
	按位或
&&	逻辑与
	逻辑或
?:	三元运算符，条件选择表达式。条件运算符“?”前的部分是判定条件，“?”和“;”间的部分为当条件为 True 时的表达式，“:”后的部分为当条件为 False 时的表达式。
=, op=	赋值、自操作与赋值 Op 可以是+, -, *, /, >>, <<, &, ^,  , %, or** =和 op=在表达式节点中不可用

## 参考文献

- [1] 陈国顺, 等. 精通 LabVIEW 程序设计 (第 2 版). 北京: 电子工业出版社, 2012.
- [2] 陈锡辉, 等. LabVIEW 8.20 程序设计从入门到精通. 北京: 清华大学出版社, 2007.
- [3] 徐晓东, 等. LabVIEW 8.5 常用功能与编辑实例精讲. 北京: 电子工业出版社, 2009.
- [4] 刘晋霞, 等. LabVIEW 2012 中文版虚拟仪器从入门到精通. 北京: 机械工业出版社, 2012.
- [5] 刘其和, 等. LabVIEW 虚拟仪器程序设计与应用. 北京: 化学工业出版社, 2011.
- [6] 王超, 等. LabVIEW 2015 虚拟仪器程序设计. 北京: 机械工业出版社, 2016.
- [7] 杨高科. LabVIEW 虚拟仪器项目开发与管理. 北京: 机械工业出版社, 2012.
- [8] 侯国屏, 等. LabVIEW7.1 编程与虚拟仪器设计. 北京: 清华大学出版社, 2005.
- [9] 王福明, 等. LabVIEW 程序设计与虚拟仪器. 西安: 西安电子科技大学出版社, 2009.
- [10] 杨乐平, 等. LabVIEW 程序设计与应用 (第 2 版). 北京: 电子工业出版社, 2005.
- [11] 德湘轶, 等. LabVIEW 程序设计基础. 北京: 清华大学出版社, 2012.
- [12] 邓奕, 等. LabVIEW 虚拟仪器程序设计与应用. 武汉: 华中科技大学出版社, 2015.
- [13] 鹏勇, 等. LabVIEW 虚拟仪器设计及分析. 北京: 清华大学出版社, 2011.

## 反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396; (010) 88258888

传 真：(010) 88254397

E-mail: [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036